BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM

ThS. Nguyễn Thị Thanh Thủy
ThS. Lâm Thị Họa Mi

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

(Lưu hành nội bộ)

THÀNH PHỐ HỒ CHÍ MINH – NĂM 2015

ThS. Nguyễn Thị Thanh Thủy
ThS. Lâm Thị Họa Mi

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

(*Tài liệu dùng cho hệ Đại học chính quy, Cao đẳng chính quy*)

THÀNH PHỐ HỒ CHÍ MINH – NĂM 2016

LỜI NÓI ĐẦU

Bài giảng Hệ quản trị cơ sở dữ liệu ra đời nhằm cung cấp cho sinh viên những kiến thức cơ bản về hệ quản trị cơ sở dữ liệu quan hệ, các chức năng và công cụ cần thiết để quản trị cơ sở dữ liệu cho người phát triển hệ thống. Với kinh nghiệm giảng dạy của bản thân, cùng với những đóng góp quý báu của đồng nghiệp trong khoa, các tác giả đã cố gắng thể hiện những nội dung trong bài giảng là những kiến thức cần thiết nhất để truyền đạt đến cho người đọc.

Bài giảng này là tài liệu chính thức và thống nhất của Bộ môn Hệ thống Thông tin, Khoa Công nghệ Thông tin và được dùng để giảng dạy cho sinh viên các hệ đại học và cao đẳng tại trường Đại học Công nghiệp thực phẩm TPHCM.

Bài giảng được chia thành các chương như sau:

Chương 1: Tổng quan về hệ quản trị cơ sở dữ liệu

Chương 2: Xây dựng và khai thác dữ liệu

Chương 3: Lập trình cơ sở dữ liệu bằng t-sql

Chương 4: Bảo mật và an toàn dữ liệu

Sau mỗi chương tác giả đã cố gắng tóm tắt cũng như đưa ra hệ thống câu hỏi và bài tập nhằm giúp cho sinh viên có thể hệ thống và tự kiểm tra trình độ tiếp nhận của bản thân. Việc biên soạn bài giảng này cũng khó tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp của quý thầy, cô và bạn đọc để việc hoàn thiện bài giảng được tốt hơn.

Nhóm tác giả

MỤC LỤC

Chương 1.
TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Mục tiêu:

- *Mô tả được các thành phần và kiến trúc SQL Server 2008.*
- *Trình bày và có thể lựa chọn được một phiên bản SQL Server 2008 để sử dụng*
- *Thực hiện được việc cài đặt SQL Server 2008.*

1.1. Giới thiệu

SQL Server 2008 là một hệ quản trị CSDL (Relational Database Management System – RDBMS) do MS phát triển. SQL Server là một hệ quản trị CSDL quan hệ mạng máy tính hoạt động theo mô hình khách – chủ (Client – Server) cho phép đồng thời cùng lúc có nhiều người dùng truy xuất đến dữ liệu, quản lý việc truy nhập hợp lệ và các quyền hạn của từng người dùng trên mạng. Ngôn ngữ truy vấn quan trọng của MS SQL Server là Transact – SQL. Transact – SQL là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institude) được sử dụng trong SQL Server.

SQL Server 2008 có nhiều phiên bản khác nhau, từ các phiên bản miễn phí như SQL Server 2008 Express cho đến SQL Server 2008 Enterprise dành cho doanh nghiệp. Các phiên bản khác bao gồm Standard, WorkGroup, Developer, Web và Compact (có thể chạy trên PC và các thiết bị sử dụng Windows Mobile), và có sự mới lạ trong các phiên bản so với SQL Server 2005 là phiên bản Web.

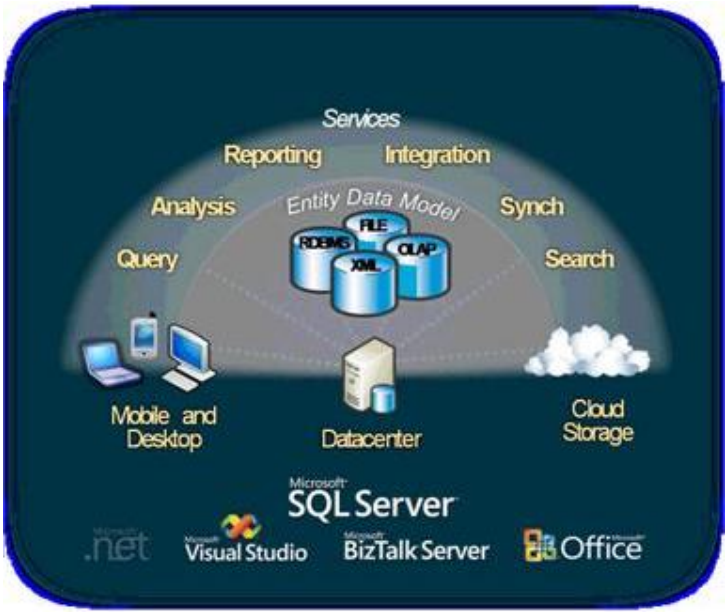
Những điểm mới trong SQL Server 2008:

- SQL Server 2008 cung cấp một nền tảng tin cậy và an toàn cho phép các tổ chức, cá nhân có thể chạy hầu hết các ứng dụng phức tạp của họ.

- SQL Server 2008 cùng với .NET Framework đã làm giảm sự phức tạp trong việc xây dựng và phát triển các ứng dụng mới. ADO.NET Entity Framework cho phép các lập trình viên có thể nâng cao khả năng làm việc với các thực thể dữ liệu logic đáp ứng được hầu hết yêu cầu của các ứng dụng thay vì lập trình trực tiếp với các bảng và cột thuần túy như trước đây. SQL Server 2008 còn hỗ trợ các hệ thống kết nối cho phép các lập trình viên xây dựng các ứng dụng mà trong đó người dùng có thể đưa dữ liệu cùng với các ứng dụng này vào những thiết bị và sau đó đồng bộ dữ liệu của chúng với máy chủ trung tâm.

- SQL Server 2008 cho phép người sử dụng khai thác triệt để và quản lý bất kỳ kiểu dữ liệu nào từ các kiểu dữ liệu truyền thống đến dữ liệu mới. Chẳng hạn so với SQL Server 2005, SQL Server 2008 có thêm những kiểu dữ liệu mới như Date, Time, …

- SQL Server 2008 cho phép người dùng dễ dàng hơn trong việc truy cập thông tin thông qua sự tích hợp sâu hơn với Microsoft Office. Điều này càng làm cho các ứng dụng công nghệ thông tin càng được phát triển rộng khắp.



Hình 1.1. Toàn cảnh nền tảng dữ liệu của Microsoft

1.2. Hệ quản trị cơ sở dữ liệu quan hệ

Một hệ quản trị cơ sở dữ liệu (HQTCSDL/ DBMS: Database Mananger System) là một hệ thống gồm một CSDL và các thao tác trên CSDL đó, được thiết kế trên một nền tảng phần cứng, phần mềm và với một kiến trúc nhất định.

1.3. Mô hình Khách – Chủ (Client – Server)

1.3.1. Khái niệm về cấu trúc vật lý

Yếu tố căn bản đầu tiên trong mô hình Client – Server (khách - chủ) là phải có hệ thống mạng các máy tính được nối kết chung với nhau theo một nghi thức nào đó. Trong hệ thống mạng cho phép kết nối hai hay nhiều máy tính bằng một sợi dây cáp với mục đích cùng sử dụng chung các tài nguyên, dữ liệu giữa các máy tính. Trong mô hình này chúng ta có các khái niệm cơ bản như sau:

- Máy chủ (Server): Là một máy tính phải có bộ xử lý với tốc độ cao (CPU), tài nguyên lớn (RAM, HardDisk) để hoạt động tốt bởi vì cùng lúc sẽ có nhiều người dùng mạng truy xuất về máy chủ thông qua các máy trạm. Trong một hệ thống mạng máy tính được phép có nhiều máy chủ với các chức năng độc lập riêng biệt khác nhau.
- Máy trạm (Client): Là các máy tính được phép truy xuất các tài nguyên đã được chia sẽ trên mạng.
- Dây cáp mạng (Cable): Là một hệ thống dây kim loại hoặc quang học nối kết vật lý các máy tính, máy in lại với nhau.

- Dữ liệu chung (Share data): Là các tập tin hay thư mục mà người sử dụng trong hệ thống mạng có thể truy xuất vào máy chủ thông qua các máy trạm và dây cáp mạng.

1.3.2. Khái niệm về các xử lý

Các xử lý trong mô hình khách – chủ phải được tổ chức sao cho hoạt động của hệ thống đạt được kết quả cao nhất. Việc tổ chức các xử lý phải đảm bảo rằng các yêu cầu (Request) từ các máy trạm phải được máy chủ phúc đáp (Response) một cách nhanh chống mà không làm tắc nghẽn hệ thống mạng máy tính. Các xử lý được chia ra làm hai nhánh khác nhau: nhánh máy trạm (Client Side) và nhánh máy chủ (Server Side).

Nhánh máy trạm (Client Side)

- Các ứng dụng bên nhánh máy trạm thường sẽ thực hiện các công việc như: đọc và hiển thị dữ liệu hiện có bên trong CSDL, tính toán dữ liệu đang hiển thị trên các màn hình ứng dụng, in dữ liệu ra các kết xuất.

- Các ngôn ngữ được dùng để xây dựng ứng dụng cho bên nhánh máy trạm thường là: Delphi, Visual basic, C++,…Các ứng dụng này còn cho phép người dùng có thể thực hiện các hành động thêm, sửa, xoá dữ liệu hiện có bên trong CSDL bên nhánh máy chủ.

- Khi đọc dữ liệu từ máy chủ về để xử lý bên nhánh máy trạm, các ứng dụng khi xây dựng nên tránh việc đọc toàn bộ dữ liệu của bảng (table) mà chỉ nên lấy về đúng các thông tin cần thiết cho các xử lý. Điều này sẽ làm giảm đi lượng thông tin lưu thông trên hệ thống mạng máy tính. Khi đó chỉ có những thông tin nào đúng với yêu cầu của nhánh máy trạm sẽ được máy chủ truyền tải trên hệ thống mạng. Ngoài ra nó cũng làm cho ứng dụng được chạy nhanh hơn là vì khi đó các xử lý chọn lựa dữ liệu được thực hiện cục bộ tại máy chủ. Với việc trang bị cấu hình cao tại máy chủ thì đảm bảo rằng việc xử lý thông tin tại máy chủ phải ở tốc độ cao nhất có thể được.

Nhánh máy chủ (Server Side)

- Các xử lý bên nhánh máy chủ sẽ được tổ chức và thực hiện trực tiếp ngay trên máy chủ. Xử lý quan trọng đầu tiên là đảm bảo việc truy cập của các người dùng trên mạng là bảo mật. Điều này có nghĩa là chỉ những người dùng nào được cấp quyền truy cập thì mới có thể truy xuất được các dữ liệu dùng chung.

- Các xử lý liên quan đến việc thực hiện hoặc cập nhật dữ liệu đồng thời cùng lúc giữa những người dùng hiện hành trên mạng.

- Các xử lý sao lưu dữ liệu (backup data) tự động để đảm bảo không mất dữ liệu khi gặp sự cố xảy ra.

1.3.3. Lợi ích việc phát triển ứng dụng mô hình Client – Server

Trước khi mô hình Client – Server ra đời, các hệ thống ứng dụng quản lý vận hành trên các loại CSDL khác vẫn hoạt động tốt, tuy nhiên phần lớn các tổ chức, công ty có xu hướng chuyển sang mô hình Client – Server vì các lý do sau:

- Giảm chi phí: với mô hình Client – Server ta có thể sử dụng các máy chủ là các máy tính cá nhân, thay vì sử dụng máy tính lớn.
- Tốc độ nhanh: việc phân chia các xử lý ra làm hai phần (nhánh máy chủ và nhánh máy trạm) sẽ làm giảm bớt việc tắc nghẽn thông tin trong hệ thống mạng máy tính. Các xử lý nào phức tạp tác động nhiều lên CSDL sẽ được lưu trữ ngay trên máy chủ, các xử lý đơn giản sẽ được thực hiện ngay trong ứng dụng bên máy trạm. Khi đó sẽ làm cho hệ thống vận hành với hiệu quả cao hơn.
- Tính tương thích cao: với mô hình Client – Server việc chọn lựa các phần mềm để phát triển ứng dụng có thể hoàn toàn độc lập từ ngôn ngữ lập trình đến hệ CSDL quan hệ và cả các thiết bị phần cứng khác. Chúng ta có thể chọn ra các thành phần tối ưu nhất theo đúng cái chúng ta cần khi xây dựng một hệ thống ứng dụng.

1.4. Các thành phần của SQL Server 2008

1.4.1. Database Engine

Đây là thành phần chính của SQL Server 2008, hoạt động với hiệu suất cao, chịu trách nhiệm quản lý, lưu trữ dữ liệu và xử lý các thao tác ảnh hưởng đến cơ sở dữ liệu.

1.4.2. Analysis Services

Analysis Services bao gồm các công cụ cho việc tạo và quản lý tiến trình phân tích trực tuyến (online analytical processing - OLAP) và các ứng dụng khai thác dữ liệu.

Đây là một dịch vụ hỗ trợ mạnh mẽ việc phân tích, khai thác những thông tin tiềm tàng bên trong của một CSDL, gồm 2 thành phần chính Online Analytical Processing (OLAP) và Data Mining.

1.4.3. Reporting Services

Reporting Services bao gồm các thành phần server và client cho việc tạo, quản lý và triển khai các báo cáo. Reporting Services cũng là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.

Đây là một giải pháp tạo báo cáo chuyên dụng dành cho các doanh nghiệp, là một web service phục vụ tạo, quản lý các báo cáo cho các ứng dụng web.

1.4.4. Integration Services

Integration Services là một tập hợp các công cụ đồ họa và các đối tượng lập trình cho việc di chuyển, sao chép và chuyển đổi dữ liệu.

1.4.5. Notification Services

Dịch vụ thông báo Notification Services là nền tảng cho sự phát triển và triển khai các ứng dụng tạo và gửi thông báo. Notification Services có thể gửi thông báo theo định thời đến hàng ngàn người đăng ký sử dụng nhiều loại thiết bị khác nhau.

1.4.6. Service Broker

Service Broker là dịch vụ mới được giới thiệu trong SQL Server 2008, cung cấp chức năng trao đổi thông điệp giữa hai instance của SQL Server, hỗ trợ xây dựng các ứng dụng phân tán nhạy cảm cần bảo mật, độ tin cậy cao và khả năng mở rộng. Service Broker sử dụng cơ chế bất đồng bộ (asynchronous) và được cài đặt một cách độc lập, giúp cho việc trao đổi thông tin được hiệu quả.

Cho phép các ứng dụng client nhận ra sự bất đồng bộ.

1.4.7. Replication Services

Dịch vụ Replication cho phép thực hiện quá trình sao chép và phân phối dữ liệu và các đối tượng của cơ sở dữ liệu từ một cơ sở dữ liệu sang nơi khác, sau đó đồng bộ giữa hai cơ sở dữ liệu để duy trì sự nhất quán dữ liệu.

Cung cấp khả năng tự động phân phối dữ liệu giữa các database hoặc giữa các server.

1.4.8. Full-Text Search

Dịch vụ tìm kiếm Full-Text Search cho phép tìm kiếm ký tự, chuỗi, cụm từ trong cơ sở dữ liệu thay vì tìm kiếm trên từng cột dữ liệu của bảng. Ngoài ra Full-text Search còn cho phép tạo chỉ mục một cách nhanh chóng và uyển chuyển để truy vấn với từ khoá trên chuỗi dữ liệu

1.5. Phân tích, đánh giá và lựa chọn phiên bản hệ quản trị

1.5.1. Các phiên bản SQL Server 2008

- ❑ Enterprise Edition
- ❑ Standard Edition
- ❑ Workgroup Edition
- ❑ Web Edition
- ❑ Express Edition
- ❑ Express Advanced Edition
- ❑ Developer Edition
- ❑ Compact Edition

Trong đó:

- SQL Server 2008 Enterprise Edition (x86, x64)

Phiên bản này được sử dụng trong các doanh nghiệp, tổ chức có các mức yêu cầu xử lý giao dịch trực tuyến trên diện rộng (online transaction processing - OLTP), khả năng phân tích dữ liệu phức tạp cao, hệ thống kho dữ liệu (data warehousing systems) và web sites. Enterprise Edition phù hợp cho các tổ chức lớn và các yêu cầu phức tạp.

- SQL Server 2008 Standard Edition (x86, x64)

Standard Edition là phiên bản phục vụ cho việc quản trị và phân tích dữ liệu phù hợp cho các doanh nghiệp, tổ chức vừa và nhỏ. Nó bao gồm các giải pháp cần thiết cho thương mại điện tử (e-commerce), kho dữ liệu (data warehousing) và dòng doanh nghiệp (line-of-business).

- SQL Server 2008 Small Business Edition (x86, x64)

Đây là phiên bản SQL Server 2008 Standard nhưng chỉ sử dụng cho tối đa 75 máy tính.

- SQL Server 2008 Developer Edition (x86, x64)

Developer Edition có tất cả các tính năng của phiên bản SQL Server 2005 Enterprise, nhưng nó chỉ là phiên bản sử dụng cho phát triển và kiểm tra ứng dụng. Phiên bản này phù hợp cho các cá nhân, tổ chức xây dựng và kiểm tra ứng dụng.

- SQL Server 2008 Workgroup Edition (x86, x64)

Workgroup Edition là giải pháp quản trị dữ liệu phù hợp cho các doanh nghiệp, tổ chức nhỏ chỉ cần một cơ sở dữ liệu không giới hạn kích thước hoặc số người sử dụng. Workgroup Edition là lý tưởng cho các mức cơ sở dữ liệu tin cậy, mạnh mẽ và dễ quản trị.

- SQL Server 2008 Web Edition (x86, x64): được thiết kế đặc biệt dành cho các ứng dụng web và giá thành rẽ hơn so với phiên bản Standard. Đây cũng là phiên bản không có giới hạn về kích thước của cơ sở dữ liệu (nếu sử dụng Express Edition thì kích thước giới hạn của cơ sở dữ liệu là 4GB).

- SQL Server 2008 Express Edition (SSE) (x86, x64)

Là hệ cơ sở dữ liệu miễn phí, dễ sử dụng và quản lý đơn giản, một số tính năng bị hạn chế so với các phiên bản khác. SQL Server Express có thể dùng như máy Client hoặc máy chủ Server đơn giản. Nó là sự lựa chọn tối thiểu nếu thích hợp cho việc học tập và xây dựng các ứng dụng nhỏ.

- SQL Server 2008 Compact 4.0 Edition (x86, x64)

Một phiên bản miễn phí cung cấp một cơ sở dữ liệu lưu trữ dễ dàng và đơn giản để sử dụng cho các ứng dụng web chạy trên máy tính để bàn hay trên thiết bị di động. Ngoài ra, trong SQL Server Compact 4.0 còn có những lệnh T-SQL mới chẳng hạn như lệnh OFFSET&FETCH, với độ tin cậy cao hơn.

Bảng 1.1. Chi tiết yêu cầu hệ điều hành của các phiên bản có thể sử dụng được:

Hệ điều hành	**Phiên bản SQL Server**					
	Enterprise	**Standard**	**Workgroup**	**Web**	**Developer**	**Express**
Windows XP SP2 Pro		x	x	x	x	x
Windows XP Home Edition SP2					x	x
Windows Server 2003 SP2 Web						x
Windows Server 2003 SP2 Standard	x	x	x	x	x	x
Windows Server 2003 SP2 Enterprise	x	x	x	x	x	x
Windows Server 2003 SP2 Datacenter	x	x	x	x	x	x
Windows Server 2008 Web	x	x	x	x	x	x
Windows Server 2008 Standard	x	x	x	x	x	x
Windows Server 2008 Enterprise	x	x	x	x	x	x
Windows Server 2008 Datacenter	x	x	x	x	x	x
Windows 7						

✍ **Nhận xét**:

Trong các phiên bản trên thì Developer Edition là sự chọn lựa phù hợp nhất cho việc học tập cơ bản và nâng cao vì ở phiên bản này hội đủ hầu hết các tính năng của phiên bản enterprise (phiên bản đầy đủ nhất), nên trong tài liệu này sẽ hướng dẫn cài đặt và sử dụng SQL Server dựa trên phiên bản Developer.

1.5.1. Cài đặt SQL Server 2008 Developer Edition

Trước khi cài đặt SQL Server 2008 cần phải chắc chắn rằng cấu hình máy tính đáp ứng được các yêu cầu đối với SQL Server 2008. Có rất nhiều các yêu cầu khác nhau về SQL Server 2008 mà máy tính phải đáp ứng, nó phụ thuộc vào phiên bản SQL Server cần cài đặt và nền tảng của hệ điều hành. Để kiểm tra điều này nên xem lại trang web sau đây để xác định các cài đặt máy đúng cho môi trường đang sử dụng: http://msdn.microsoft.com/en-us/library/ms143506.aspx

✍ **Lưu ý:** Một số yêu cầu tối thiểu để cài đặt SQL Server 2008 phiên bản Developer Edition (tham khảo)

Yêu cầu về phần cứng:

- Processor: Pentium IV 2Gb MHz trở lên.
- Ram: 1GB trở lên.
- Hard Disk: 10GB.

Yêu cầu về phần mềm:

- Hệ điều hành tối thiểu: Windows XP SP3, Windows Vista SP1, Windows 7, Windows 2003 SP2.
- Net Framework 3.5
- Windows Installer 4.5 trở lên.

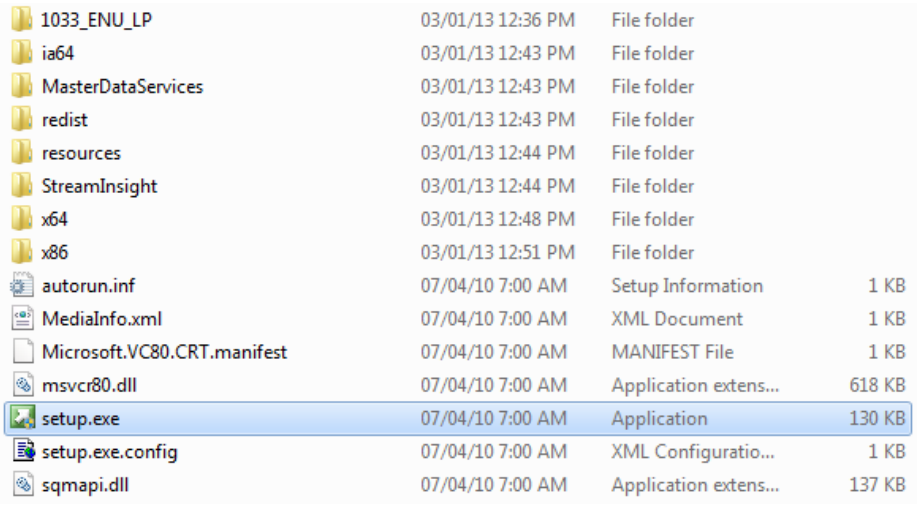
Các bước thực hiện cài đặt SQL Server 2008 Developer Edition

Bước 1: Cài đặt Windows Installer 4.5

Nếu hệ thống chưa cài đặt Windows Installer 4.5 thì tiến hành cài đặt Windows Installer 4.5 hoặc cao hơn.

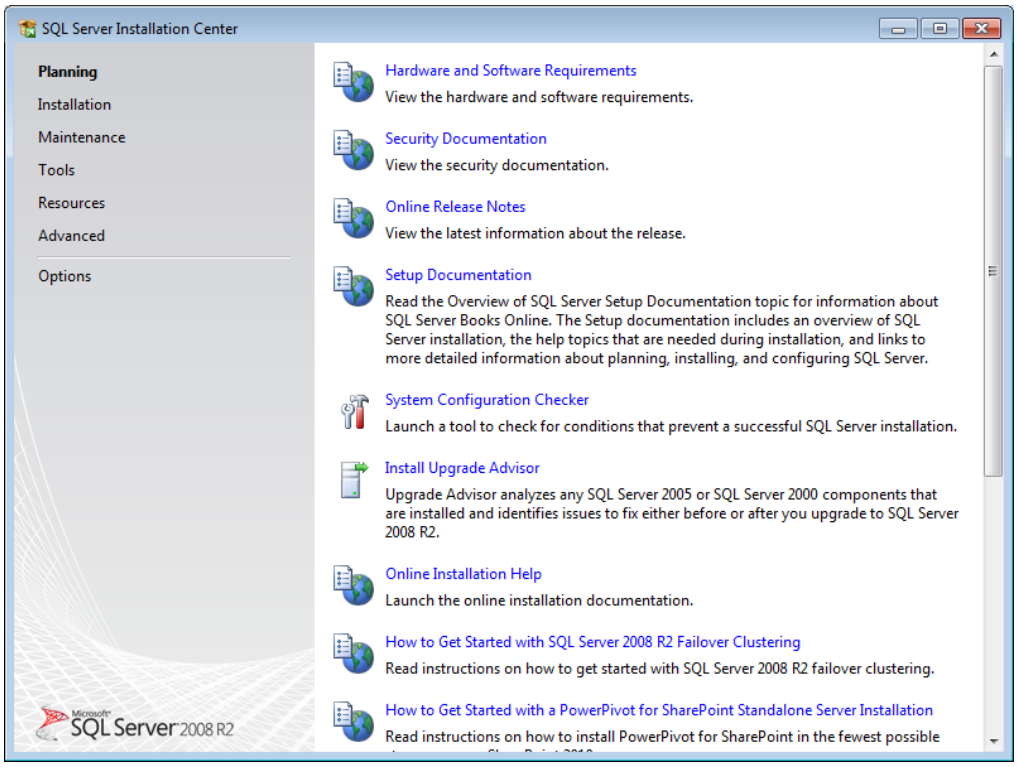
Bước 2: Cài đặt SQL Server

Cho đĩa chương trình cài đặt SQL Server 2008 vào ổ đĩa, nó sẽ tự động khởi động quá trình thiết lập cho SQL Server 2008. Nếu cài đặt không bắt đầu có thể tìm tập tin "**setup.exe**" và khởi động nó.



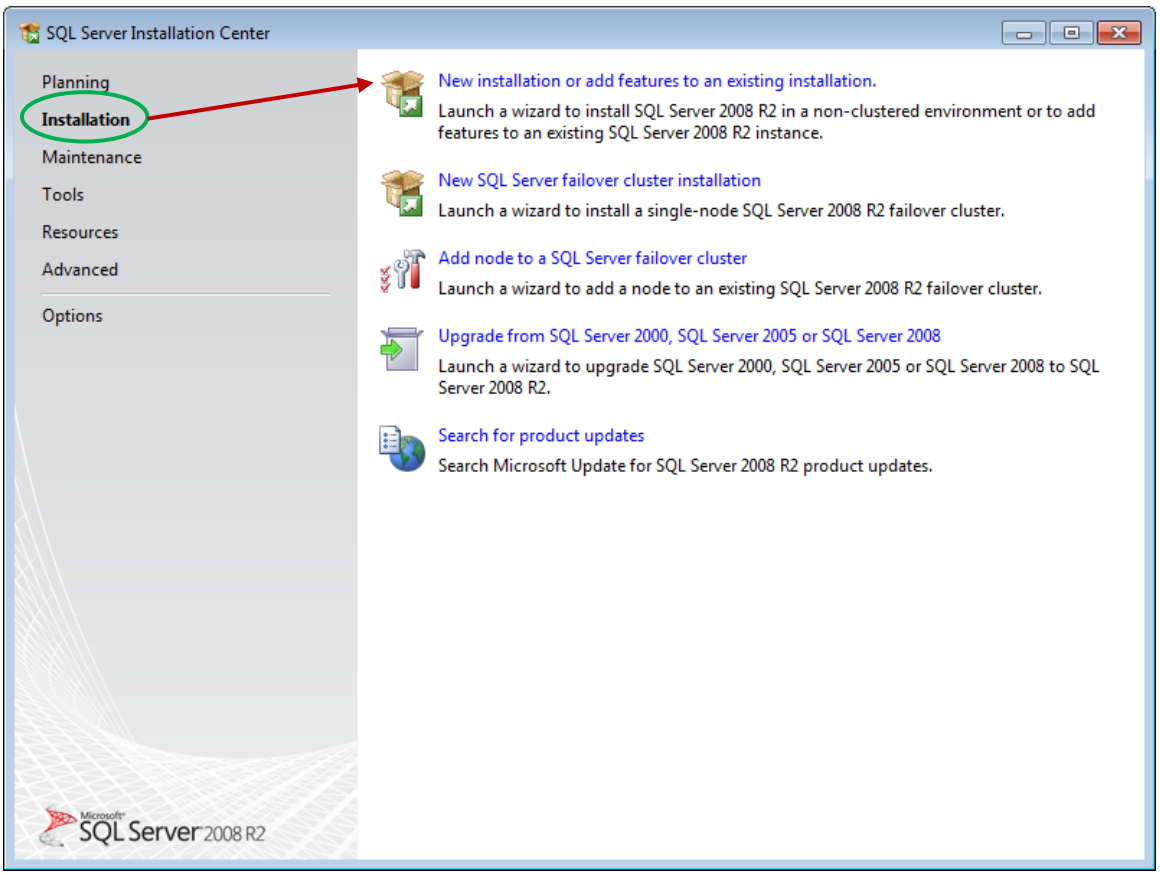
Hình 1.2. Thư mục chức tập tin cài đặt SQL Server

Quá trình cài đặt bắt đầu với những thiết lập ban đầu như sau:



Hình 1.3. Giao diện SQL Server Installation Center

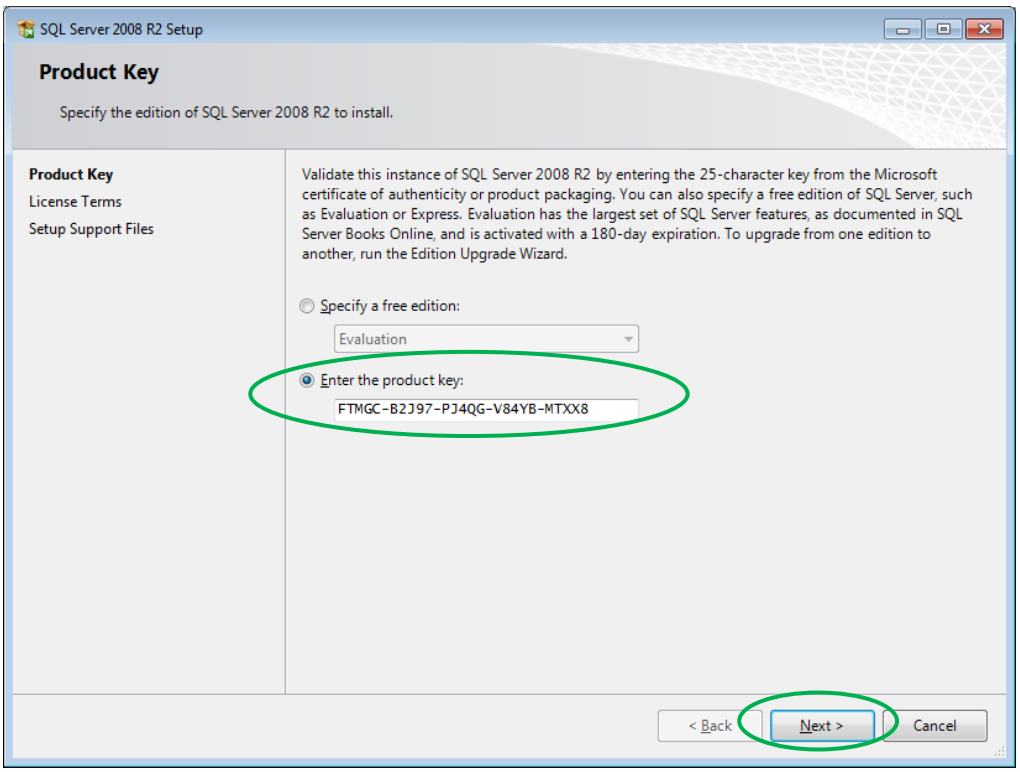
Trong cửa sổ ở hình 1.3 có thể thấy có một số tùy chọn khác nhau. Để bắt đầu quá trình cài đặt cần phải chọn *Installation* ở khung bên trái, khi đó màn hình bên dưới sẽ hiển thị:



Hình 1.4. Giao diện SQL Server Installation Center

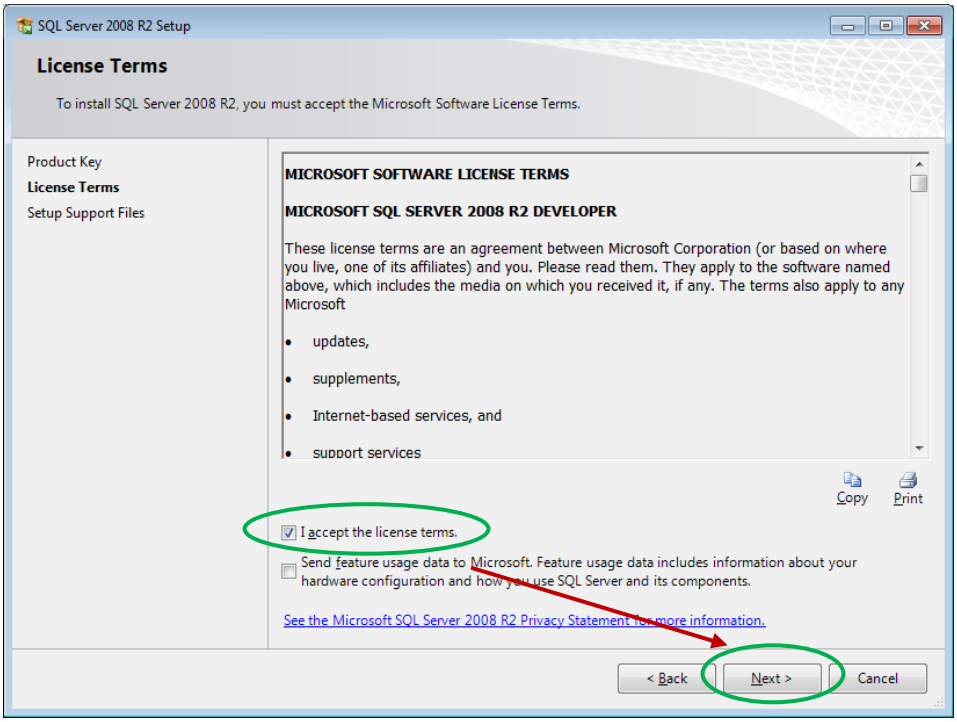
Chọn *New installation or add features to …* → Chọn *OK*.

Cửa sổ *Product Key* hiển thị yêu cầu cung cấp Product Key. Chọn phiên bản phù hợp, nhập key và sau đó click vào nút "**Next**".



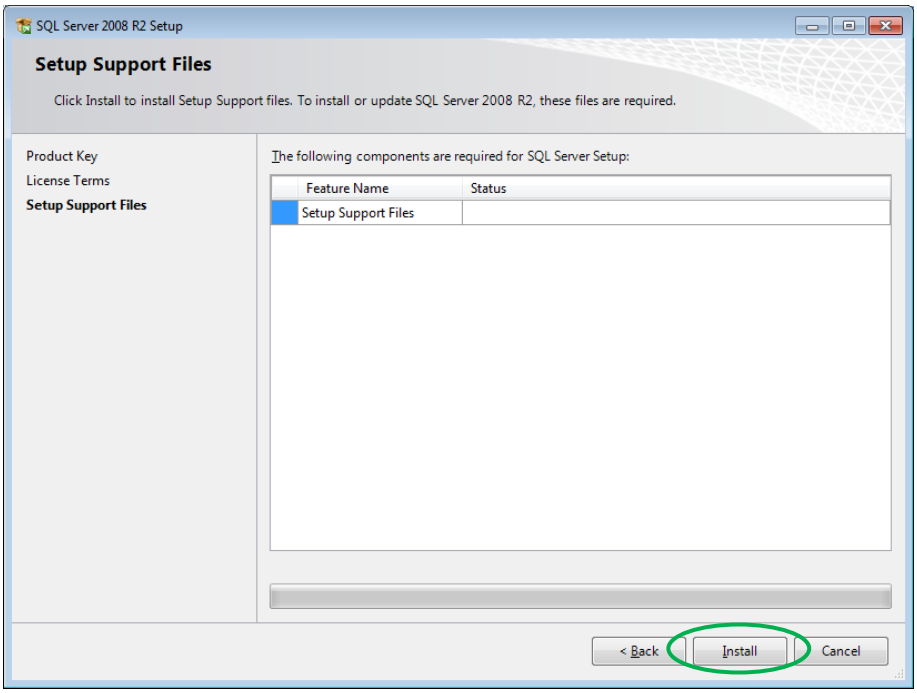
Hình 1.5. Giao diện nhập Product Key

Cửa sổ hiển thị điều khoản về giấy phép "*License Terms*". Nếu đồng ý với các điều khoản cấp phép thì click vào ô chấp nhận và ấn nút "**Next**" để chuyển sang quá trình cài đặt tiếp theo.



Hình 1.6. Giao diện License Terms

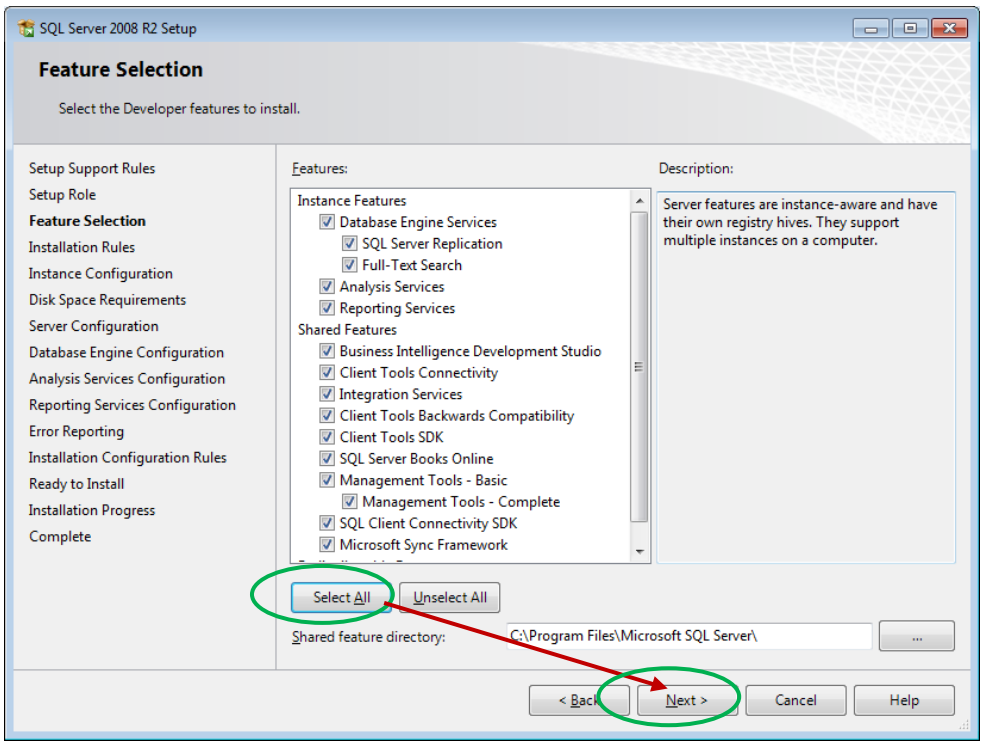
Màn hình nhắc nhở để cài đặt các thành phần được yêu cầu. Click vào nút "**Install**" để bắt đầu.



Hình 1.7. Giao diện Setup Support Files

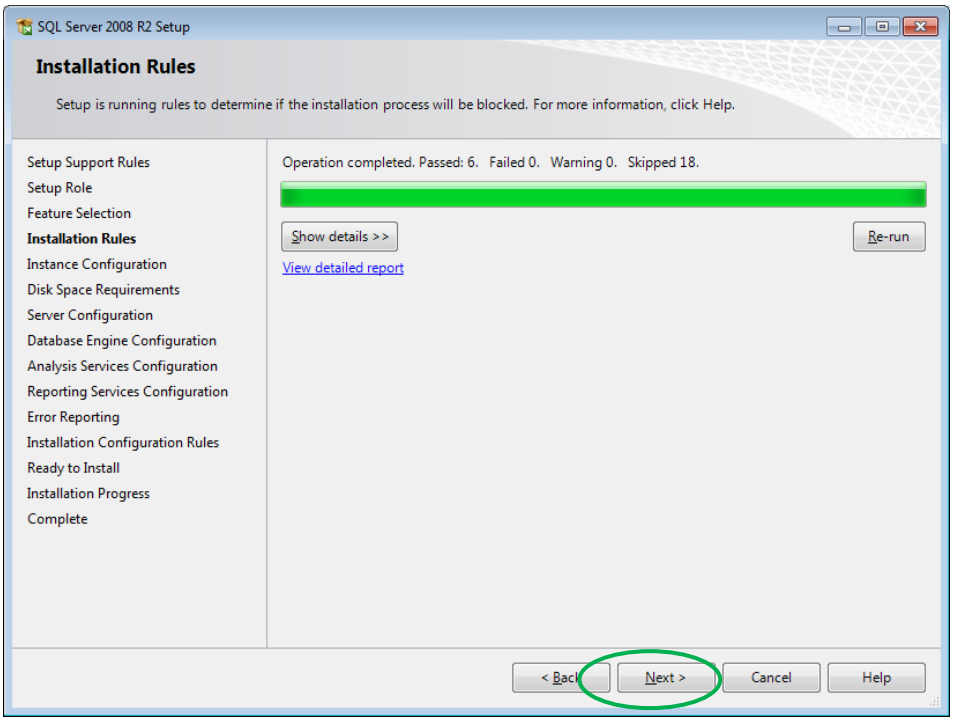
Sau khi tất cả các tập tin được cài đặt, một bản tóm tắt của quá trình thiết lập cài đặt sẽ được đưa ra. Nếu tất cả các điều kiện cần thiết đã có đủ, click Next để quá trình cài đặt bắt đầu; nếu không cần phải khắc phục những mục còn thiếu và quay lại việc cài đặt thiết lập.

Một khi đã sẵn sàng cài đặt click vào nút "**Next**". Một cửa sổ hiện ra:



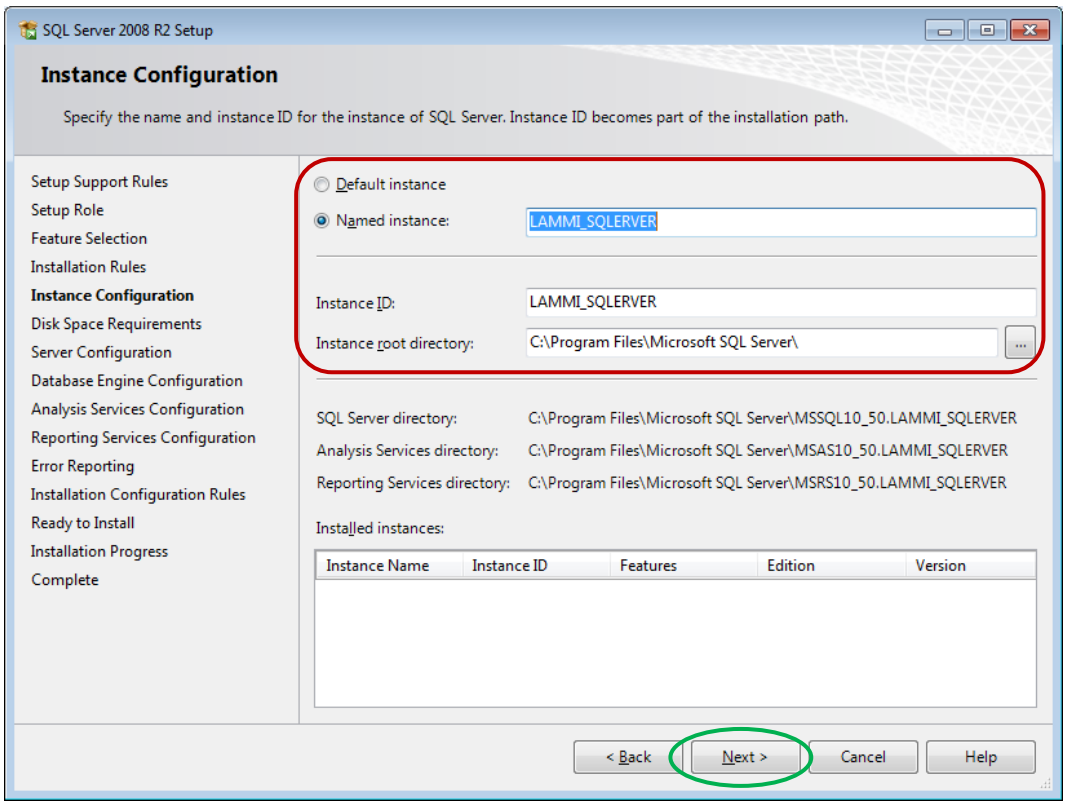
Hình 1.8. Giao diện Feature Selection

Cửa sổ màn hình ở hình 1.8 hiển thị cho phép chọn các tính năng khác nhau của SQL Server 2008 để cài đặt. Nếu muốn cài đặt đầy đủ các tính năng có thể chọn "**Select All**" → sau đó chọn "**Next**" để tiếp tục.



Hình 1.9. Giao diện Installation Rules

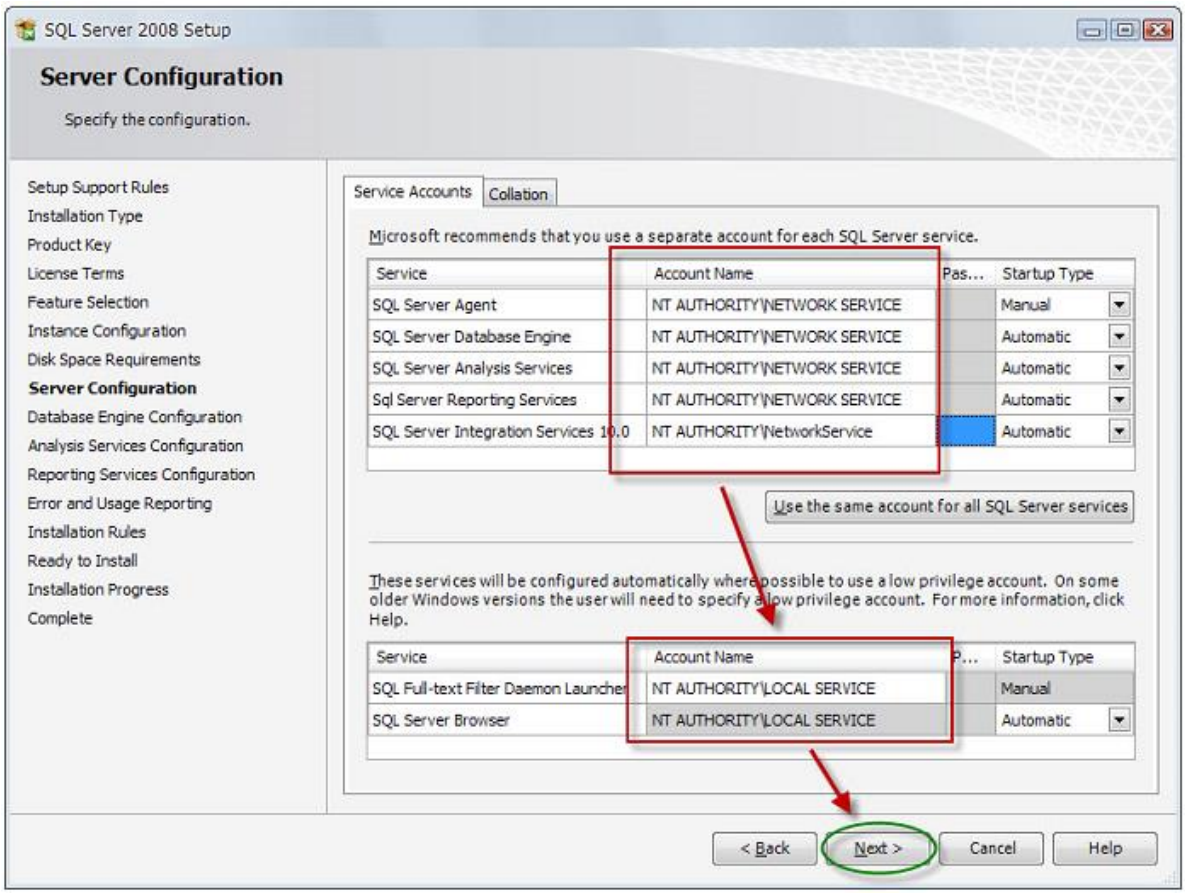
Click "**Next**". Cửa sổ sau sẽ hiện ra:



Hình 1.10. Giao diện Instance Configuration

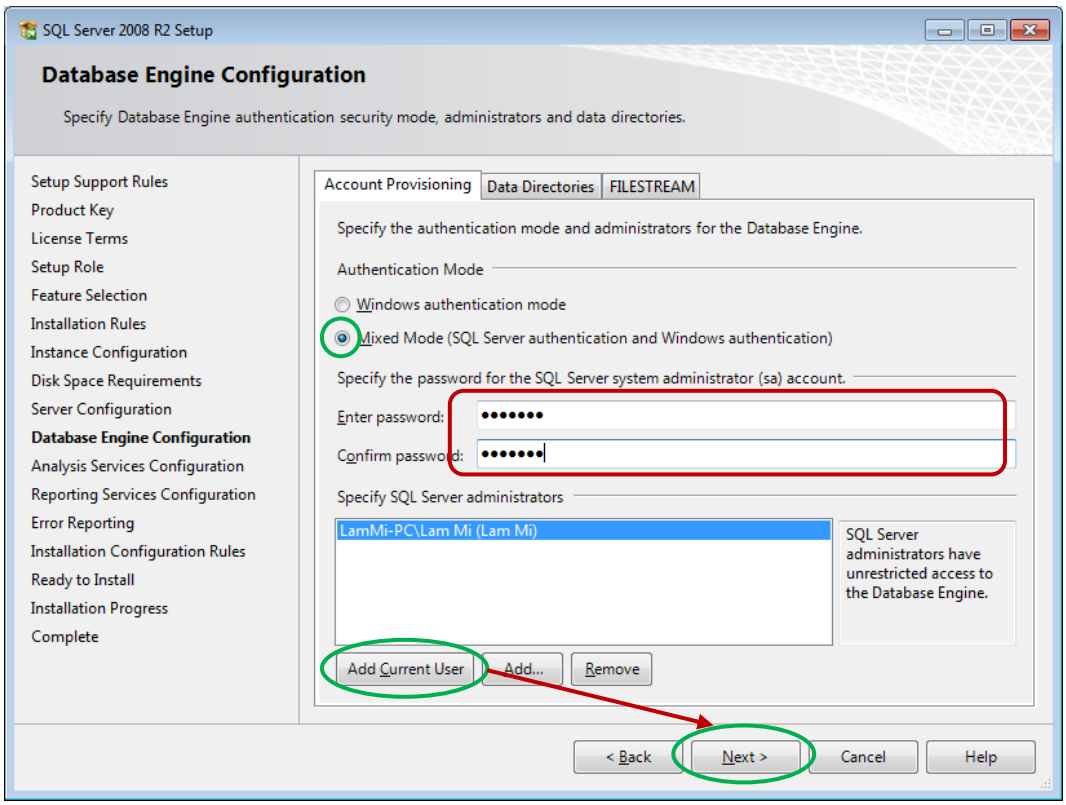
Cửa sổ trên cho phép người sử dụng thiết lập một *Named instance* cho riêng mình (nếu không muốn thiết lập *Named instance* riêng thì có thể chọn *Default instance*).

Click vào nút "**Next**". Quá trình cài đặt tiếp theo sẽ tính toán lại dung lượng còn trống trên ổ đĩa có đủ để thực hiện cài đặt hay không. Nếu không đủ, cần phải chỉ định một ổ đĩa khác để cài đặt, sử dụng nút "**Back**" để quay trở lại và chọn lại vị trí cài đặt mới. Click vào nút "**Next**" để bắt đầu, khi đó cửa sổ mới sẽ hiện ra:



Hình 1.11. Giao diện Server Configuration

Trong cửa sổ trên có thể sử dụng tab "**Server Accounts**" chỉ định tài khoản sẽ sử dụng để chạy các dịch vụ khác (services) của SQL Server 2008 và sử dụng thẻ "**Collation**" để chỉ định dãy hòa trộn muốn sử dụng. Trong hình trên đã thiết lập một tài khoản giống nhau cho tất cả dịch vụ của SQL Server. Để thực hiện người sử dụng có thể vào tài khoản đó nhiều lần với mỗi dịch vụ hoặc có thể click vào "**Use the same account for all SQL Server services**" và chỉ nhập tài khoản và mật khẩu một lần duy nhất. Click vào nút "**Next**" để quá trình cài đặt được tiến hành. Cửa sổ mới sẽ hiện ra như sau:



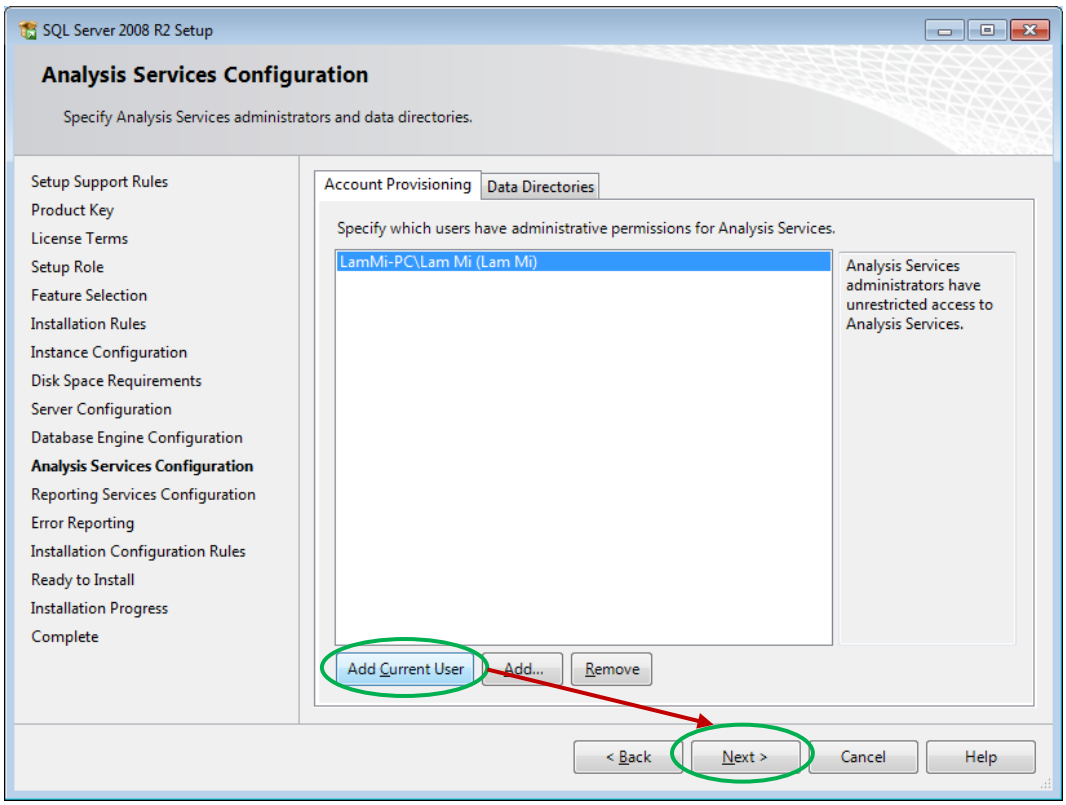
Hình 1.12. Giao diện Database Engine Configuration

Trong màn hình trên, chúng ta tiến hành chọn chế độ xác thực của SQL Server. Có 2 chế độ xác thực:

- Windows authentication mode: chế độ xác thực bằng tài khoản của Windows.
- Mixed mode: chế độ xác thực hỗn hợp (Xác thực bằng tài khoản của Windows hoặc bằng tài khoản trong SQL Server).

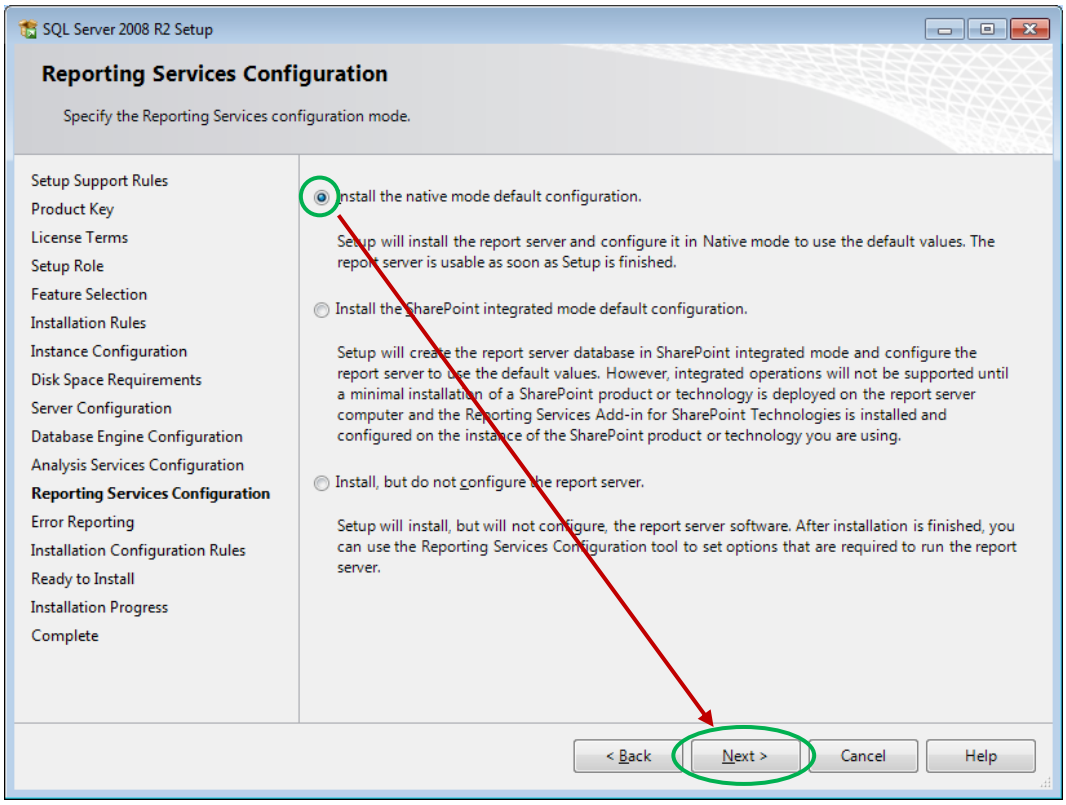
Nếu chọn chế độ xác thực hỗn hợp thì phải đặt mật khẩu cho tài khoản '**sa**'.

Màn hình *Analysis Services Configuration* → chọn "Add Current User" (sử dụng người dùng hiện tại)→ chọn "Next" để tiếp tục.

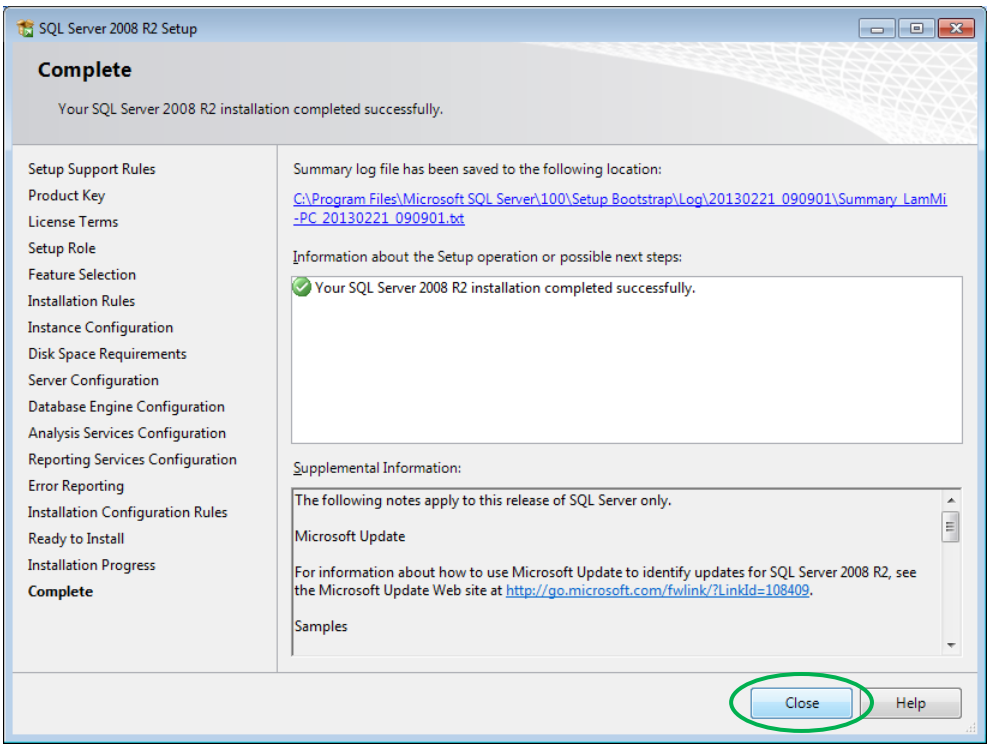


Hình 1.13. Giao diện Analysis Services Configuration

Màn hình *Reporting Services Configuration*, chọn option như hình, chọn Next, Next, … cho đến khi hoàn tất.



Hình 1.14. Giao diện Reporting Services Configuration



Hình 1.15. Giao diện hoàn thành chương trình cài đặt SQL Server 2008

Click vào nút "Close" quá trình cài đặt SQL Server 2008 thành công.

Kết chương

Chúng ta vừa tìm hiểu tổng quan về hệ quản trị cơ sở dữ liệu, kiến trúc và hoạt động của mô hình Client – Server, khảo sát hoạt động ở nhánh máy trạm và máy chủ. Các thành phần hoàn toàn mới được đưa vào SQL Server 2008 so với các phiên bản trước như: Service Broker, Reporting Service, đặc biệt nâng cao hiệu năng xử lý trực tuyến với khối lượng dữ liệu lớn, tính sẵn sàng cao đáp ứng được yêu cầu ngày càng cao của người sử dụng. Ngoài ra SQL Server 2008 có nhiều phiên bản để lựa chọn cài đặt nhằm phù hợp với quy mô hoạt động của từng cá nhân, tổ chức và doanh nghiệp.

Câu hỏi và bài tập chương 1

1. Trình bày đặc điểm của hệ quản trị cơ sở dữ liệu quan hệ, cho biết một số hệ quản trị cơ sở dữ liệu thông dụng hiện nay. So sánh với hệ quản trị cơ sở dữ liệu SQL Server 2008.
2. SQL Server 2008 có những dịch vụ nào?
3. Trình bày ý nghĩa và hoạt động của các dịch vụ Service Broker, Reporting Service, Full – Text Search, Replication.
4. Trình bày tính năng của các phiên bản cài đặt của SQL Server 2008.
5. Thực hiện cài đặt SQL Server 2008 phiên bản Developer Edition.

Chương 2
XÂY DỰNG VÀ KHAI THÁC DỮ LIỆU

***Mục tiêu*:**

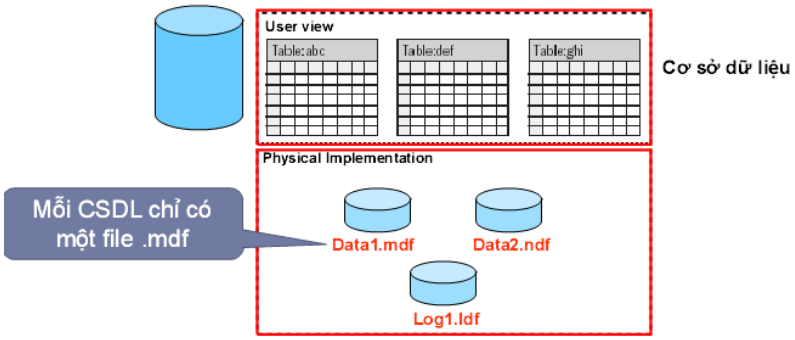
- *Trình bày được cấu trúc một cơ sở dữ liệu.*
- *Trình bày được cấu trúc của một bảng.*
- *Thực hành tạo một cơ sở dữ liệu và bảng.*
- *Thực hành tạo một lược đồ quan hệ.*

2.1. Cơ sở dữ liệu (Database)

2.1.1. Cấu trúc cơ sở dữ liệu

SQL Server lưu trữ dữ liệu trong các tập tin dữ liệu (*data files*) và lưu những thay đổi trong các tập tin log (*log files* – còn gọi là tập tin nhật ký). Các tập tin này được nhóm lại thành một cấu trúc logic được gọi là cơ sở dữ liệu. Một cơ sở dữ liệu SQL Server có thể có nhiều tập tin dữ liệu và nhiều tập tin log, mặc dù chỉ cần một tập tin log là đủ.

Khi một cơ sở dữ liệu được tạo ra, nó sẽ có duy nhất một tập tin dữ liệu chính (*primary data file*) với tên mở rộng mặc định là *.mdf* và cũng có thể tùy chọn nhiều tập tin phụ (*secondary data file*) có tên mở rộng mặc định là *.ndf*. Những tập tin dữ liệu này có thể được nhóm lại với nhau trong một nhóm logic gọi là *filegroup*. Mỗi cơ sở dữ liệu có ít nhất một tập tin *transaction log* có tên mở rộng là *.ldf*. Như vậy mỗi cơ sở dữ liệu sẽ có ít nhất hai tập tin, một tập tin *.mdf* và một tập tin *.ldf*.



Hình 2.1. Cấu trúc một cơ sở dữ liệu

2.1.1.1. Tập tin dữ liệu (Data File)

Tập tin dữ liệu trong SQL Server có 2 loại là tập tin dữ liệu chính (*primary data file*), tập tin dữ liệu phụ (*secondary data file*) và chúng có cấu trúc giống hệt nhau.

Primary data file là file chính lưu trữ dữ liệu, lưu trữ điểm bắt đầu của một CSDL và các điểm kết nối đến các file lưu trữ tiếp theo.

Secondary data file là tập tin lưu trữ dữ liệu sau primary data file, mỗi CSDL có thể có nhiều tập tin secondary. Loại tập tin này cho phép một CSDL có thể phân tán dữ liệu ở nhiều nơi trên máy tính hoặc trên mạng

Cả 2 loại file này được dùng để lưu trữ dữ liệu (*data)* cũng như tất cả các siêu dữ liệu (*metadata*). Tất cả các dữ liệu từ các bảng, index và siêu dữ liệu được tổ chức lưu trữ trong các đối tượng gọi là các Page và Extent.

Page:

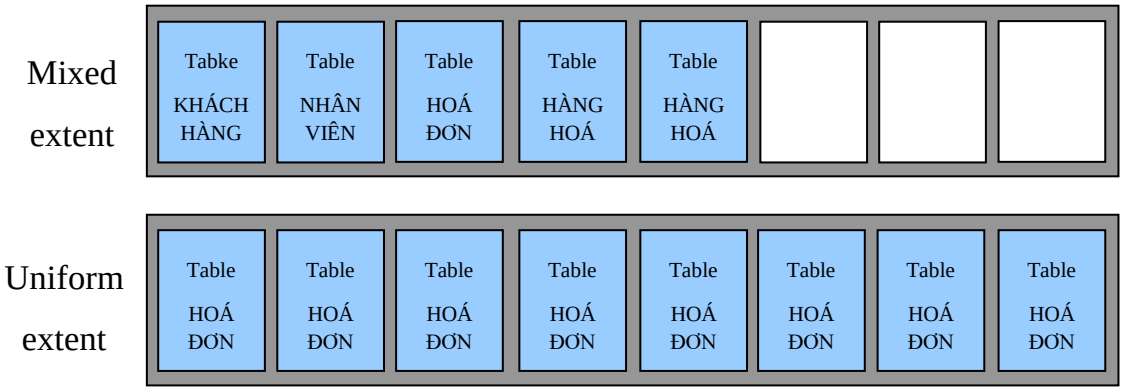
SQL Server quản lý một page có kích thước là 8KB, như vậy 1MB có 128 page. Trong mỗi trang có 96 byte chứa thông tin của trang.

Dữ liệu trong một trang sẽ bắt đầu lưu trữ sau phần thông tin Header, và lưu trữ liên tiếp, mỗi hàng có kích thước tối đa là 8060 byte. Đối với các kiểu dữ liệu phức tạp, có kích thước lớn, SQL Server sẽ tổ chức quản lý theo hình thức phân trang nhằm tăng hiệu quả truy vấn dữ liệu.

Extent:

Extent là một đơn vị lưu trữ dữ liệu trong SQL Server. Mỗi extent có kích thước là 64KB gồm 8 trang (*page*) liên tiếp. Có 2 loại extent là *mixed extent* (loại hỗn hợp) và *uniform extent* (loại thuần nhất).

- Mixed extent có thể lưu trữ nhiều đối tượng khác nhau.
- Uniform extent chỉ lưu trữ duy nhất một đối tượng.



Hình 2.2. Cấu trúc một Extent

Khi tạo mới một đối tượng (*table, index,…*) SQL Server sẽ cấp phát một mixed extent với một trang trống và cấp trang đó cho đối tượng được tạo, mỗi trang chỉ dành để chứa một đối tượng. Khi một đối tượng yêu cầu thêm không gian lưu trữ, SQL Server sẽ cấp phát thêm không gian từ các mixed extent cho đến khi đối tượng sử dụng hết 8 trang thì SQL Server sẽ cấp phát một uniform extent thay cho các trang đã cấp phát.

2.1.1.2. Tập tin nhật ký (transaction log)

Mỗi database đều có ít nhất một tập tin transaction log có phần mở rộng là *.ldf*. Mỗi tập tin transaction log đều có một tên logic dùng trong các câu lệnh T-SQL và có một tên vật lý dùng trong hệ điều hành. Không giống như tập tin dữ liệu (gồm các trang), tập tin transaction log chứa một chuỗi các mẫu tin (*record*).

Mục đích của transaction log là ghi lại thông tin tất cả các thay đổi xảy ra trong cơ sở dữ liệu. Trong cấu hình mặc định của cơ sở dữ liệu, transaction log giữ một mẫu tin về tất cả các thay đổi trên cơ sở dữ liệu và không bao giờ xóa trừ khi nó được sao lưu hoặc cắt ngắn bởi một quản trị viên.

Hoạt động của tập tin transaction log:

Mỗi khi có sự thay đổi dữ liệu trên cơ sở dữ liệu như Insert, Update, Delete được yêu cầu từ các ứng dụng, SQL Server sẽ tải (load) các trang dữ liệu tương ứng lên vùng nhớ tạm (buffer cache), sau đó sự thay đổi dữ liệu diễn ra trong buffer cache (những trang bị thay đổi còn gọi là trang dirty), và mọi sự thay đổi này đều được ghi vào tập tin transaction log trước khi ghi xuống đĩa cho nên người ta gọi là write-ahead transaction log. Hành động mà SQL Server ghi trang dirty xuống đĩa được gọi là làm sạch trang (flushing the page).

Các trang bị thay đổi sẽ được lưu xuống đĩa khi lệnh checkpoint được thi hành. Lệnh này được hệ thống tự động thực thi theo định kỳ hoặc xảy ra khi:

- Lệnh check point được gọi thực thi (cú pháp: checkpoint)
- Lệnh Alter database được thi hành
- Một thể hiện SQL Server bị dừng bất thường
- Sao lưu database

2.1.2. Xây dựng cơ sở dữ liệu

✍ **Lưu ý:**

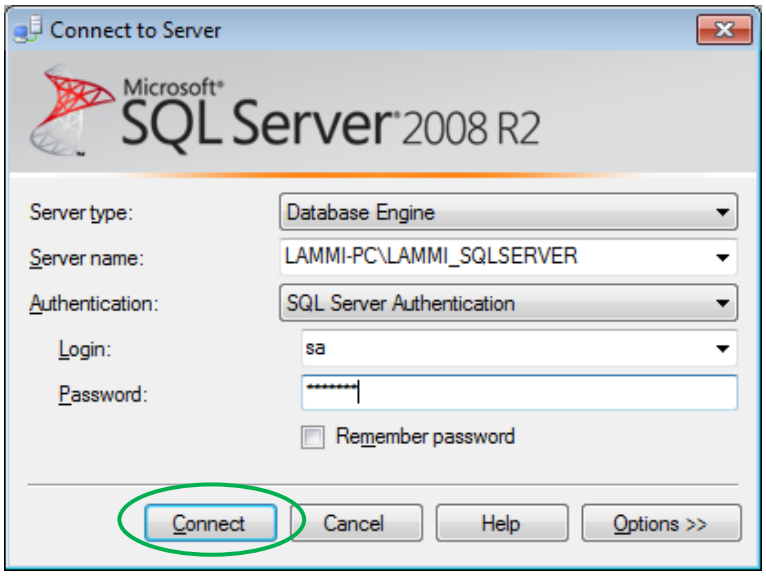
- Thông thường để tránh xảy ra rủi ro người ta thường lưu các tập tin log ở ổ đĩa khác với ổ đĩa lưu trữ các tập tin dữ liệu (data file).
- Cần dự tính dung lượng dữ liệu ban đầu cho database sử dụng đủ trong một khoảng thời gian nhất định (dung lượng cho data file và dung lượng cho transaction log file)
- Để tạo một CSDL các thông tin sau phải được yêu cầu:
  - Tên của CSDL
  - Kích thước ban đầu của CSDL
  - Các tập tin và các nhóm tập tin để lưu CSDL.

Có thể tạo một CSDL bằng:

- Dùng giao diện SQL Server Manangement Studio.
- Dùng lệnh T–SQL.

2.1.2.1. Xây dựng cơ sở dữ liệu dùng SQL Server Management Studio

Khởi động hệ quản trị CSDL SQL Server 2008 bằng cách: Start → All Progarms → Microsoft SQL Server 2008 → SQL Server Management Studio



Hình 2.3. Màn hình kết nối SQL Server 2008

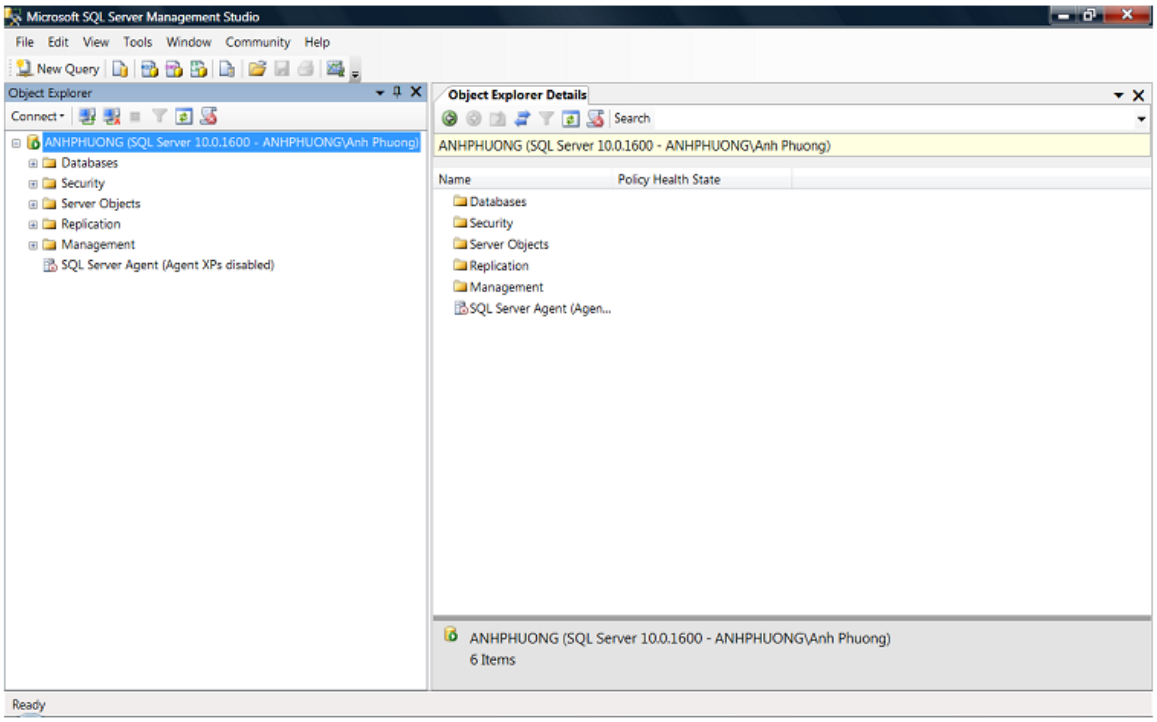
Trong đó:

Server type: trong khuôn khổ quyển giáo trình này, chúng tôi chọn Server type là *Database Engine*. Các tùy chọn khác là kiểu dữ liệu khác nhau của server nó sẽ hiển thị kết nối.

Server name: Chứa một danh sách của SQL Server đã được thiết lập khi cài đặt. Nếu bạn mở hộp Server name, bạn có thể tìm kiếm nhiều server local hoặc network connection bằng cách chọn ⟨Browse for more…⟩

Authentication: Xác định loại hình kết nối bạn muốn sử dụng. Bạn có thể sử dụng quyền kết nối Windows Authentication. Nếu bạn cài đặt SQL Server với chế độ hỗn hợp (Mix mode), thì bạn có thể chọn tùy chọn SQL Server Authentication. Khi đó nó sẽ mở hai hộp thoại cho phép bạn nhập username và password.

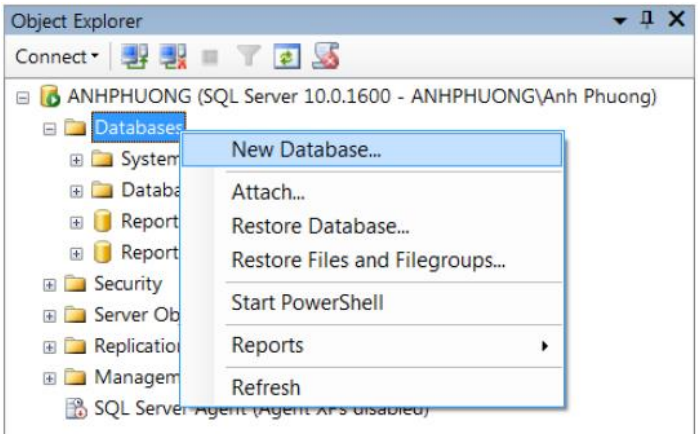
Sau khi kết nối bằng cách nhấn nút "**Connect**" màn hình sau sẽ hiện ra:



Hình 2.4. Màn hình SQL Server Management Studio

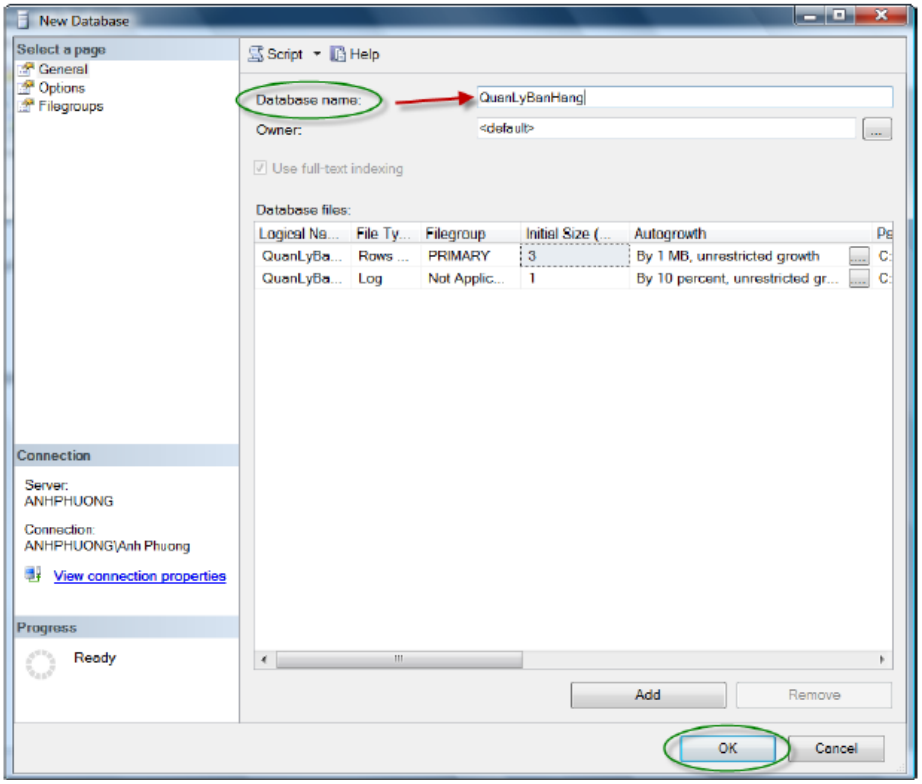
- **Các bước tạo một cơ sở dữ liệu (database)**

Bước 1: Chọn *Database* → Click phải → Chọn *New Database*…



Hình 2.5. Hộp thoại Object Explorer

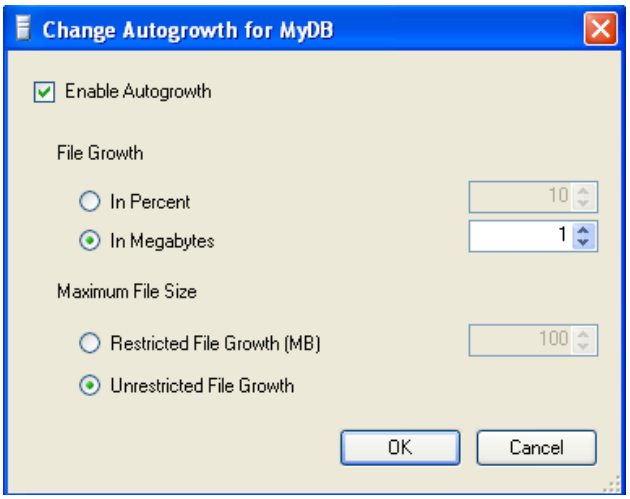
Bước 2: Trên cửa sổ *New Database* → chọn trang *General* → tại mục *Database Name* đặt tên cho database cần tạo.



Hình 2.6. Giao diện New Database

Tại vùng *Database Files* mặc định có 2 file (một file dữ liệu chính thuộc nhóm *primary* có dung lượng khởi tạo là 3MB và một file *transaction log* có dung lượng khởi tạo là 1MB). Kích thước khởi tạo này có thể được thay đổi cho phù hợp với nhu cầu của người dùng. Ngoài ra có thể tạo thêm các file dữ liệu phụ và tập tin *Log* bằng cách nhấn chọn nút *Add*.

Khi chọn xong kích thước khởi tạo, cần thiết lập tỉ lệ tăng trưởng tại cột *Autogrowth*. Mặc định cho data file là 1MB, *Transaction Log* là 10%, có thể thay đổi tỉ lệ này bằng cách nhấp chuột vào nút ... ở cuối cột *Autogrowth*.



Hình 2.7. Hộp thoại Change Autogrowth for …

Sau khi thiết lập xong các tuỳ chọn, nhấp chọn *OK* hệ thống sẽ tiến hành tạo cơ sở dữ liệu theo yêu cầu của bạn.

2.1.2.2. Xây dựng cơ sở dữ liệu bằng lệnh T-SQL

Ngoài cách tạo bằng công cụ ở trên, ta có thể thực hiện bằng lệnh T-SQL theo cú pháp như sau:

Cú pháp:

```
CREATE DATABASE <Tên_database>
ON PRIMARY
(  name = 'tên logic của file dữ liệu 1',
   Filename = 'đường dẫn đến nơi lưu trữ trên đĩa',
   Size = <dung lượng file>[KB|MB|GB|TB],
    Maxsize = <dung lượng tối đa [KB |MB |GB |TB]
                                          |UNLIMITED>,
    Filegrowth = <tỉ lệ tăng trưởng>
)[,
Filegroup <tên_nhóm_tập_tin>
(  name = 'tên logic của file dữ liệu 2',
   Filename = 'đường dẫn đến nơi lưu trữ trên đĩa',
    Size = <dung lượng>[KB|MB|GB|TB],
    Maxsize = <dung lượng tối đa [KB |MB |GB |TB]
                                          |UNLIMITED>,
    Filegrowth = <tỉ lệ tăng trưởng>
)…]
LOG ON
(  name = 'tên logic của file transaction log',
  Filename = 'đường dẫn đến nơi lưu trữ trên đĩa',
  Size = <dung lượng>[KB|MB|GB|TB],,
  Maxsize = <dung lượng tối đa [KB |MB |GB |TB]
                                          |UNLIMITED>,
  Filegrowth = <tỉ lệ tăng trưởng> )
```

Ví dụ 1: Tạo một CSDL có tên là DB_Vidu dành ra 20MB lúc đầu cho phần dữ liệu và 5MB cho phần nhật ký. Các tập tin có thể phát triển lên đến 100MB cho phần dữ liệu và 15MB đối với nhật ký.

```
CREATE DATABASE DB_Vidu
ON PRIMARY
(
    Name = DBVD_Primary,
    --chọn nơi lưu trữ và đặt tên cho tập tin primary
    Filename= 'D:\DBVD_Primary.mdf',
    Size = 20MB,
    Maxsize=100MB
)
LOG ON
(
    Name= DBVD_Log,
    --chọn nơi lưu trữ và đặt tên cho tập tin log
    Filename='D:\DBVD_Log.ldf',
    Size= 5MB,
    Maxsize= 15MB
)
```

✍ **Lưu ý**: Nếu ta không chỉ định thuộc tính nào trong cấu trúc tạo một database thì thuộc tính đó sẽ được gán giá trị mặc định. Chẳng hạn trong ví dụ 1, thuộc tính Filegrowth không được chỉ định sẽ được lấy giá trị mặc định cho file primary là 1MB và 10% cho file log.

Ví dụ 2: Tạo một cơ sở dữ liệu có tên là DB_NHANVIEN gồm:

Một tập tin chính có tên logic là DB_PRIMARY, tên lưu trữ vật lý trên đĩa là db_primary.mdf, kích thước ban đầu là: 2MB, tỉ lệ tăng trưởng là 10%, kích thước tối đa là 5MB.

Tập tin phụ thứ nhất có tên logic là 'DB_SECOND1_1, tên lưu trữ vật lý trên đĩa là DB_second1_1.ndf, kích thước ban đầu là: 2MB, tỉ lệ tăng trưởng là 10%, kích thước tối đa là 5MB.

Tập tin phụ thứ hai có tên logic là 'DB_SECOND1_2, tên lưu trữ vật lý trên đĩa là DB_second1_2.ndf, kích thước ban đầu là: 2MB, tỉ lệ tăng trưởng là 10%, kích thước tối đa là 5MB.

Tập tin phụ thứ ba có tên logic là 'DB_SECOND1_3, tên lưu trữ vật lý trên đĩa là DB_second1_3.ndf, kích thước ban đầu là: 1MB, tỉ lệ tăng trưởng là 5%, kích thước tối đa là 3MB.

Tập tin phụ thứ nhất và thứ 2 thuộc nhóm tập tin (file group) có tên là: nhom1, tập tin phụ thứ 3 thuộc nhóm tập tin có tên là: nhom2.

Tập tin transaction log có tên logic là: DB_Log, tên lưu vật lý lưu trữ trên đĩa là DB_Log.ldf, kích thước ban đầu là: 3MB, tỉ lệ tăng trưởng là 15%, kích thước tối đa là 10MB.

```
CREATE DATABASE DB_NHANVIEN
ON PRIMARY
(
    name='DB_PRIMARY',
    filename='D:\db_primary.mdf',
    size=2MB,
    maxsize=5MB,
    filegrowth=10%
),
FILEGROUP nhom1
(
    name='DB_SECOND1_1',
    filename='D:\DB_second1_1.ndf',
    size=2MB,
    maxsize=5MB,
    filegrowth=10%
),
(
    name='DB_SECOND1_2',
    filename='D:\DB_second1_2.ndf',
    size=2MB,
```

```
    maxsize=5MB,
    filegrowth=10%
),
FILEGROUP nhom2
(
    name='DB_SECOND1_3',
    filename='D:\DB_second1_3.ndf',
    size=1MB,
    maxsize=3MB,
    filegrowth=5%
)
LOG ON
(
   name='DB_Log',
   filename='D:\DB_log.ldf',
   size=3MB,
   maxsize=10MB,
   filegrowth=15%
)
```

✍ **Lưu ý**: Nếu không chỉ định một transaction log file thì SQL sẽ tự động tạo ra một log file với kích thước ban đầu là bằng ¼ dung lượng file *.mdf* và được lưu trữ cùng nơi với file .mdf.

```
CREATE DATABASE DB_SINHVIEN
ON PRIMARY
(
    name = DBSV_Primary,
    Filename= 'D:\DBSV_Primary.mdf',
    Size = 20MB,
    Maxsize=100MB,
    Filegrowth= 10MB
);
```

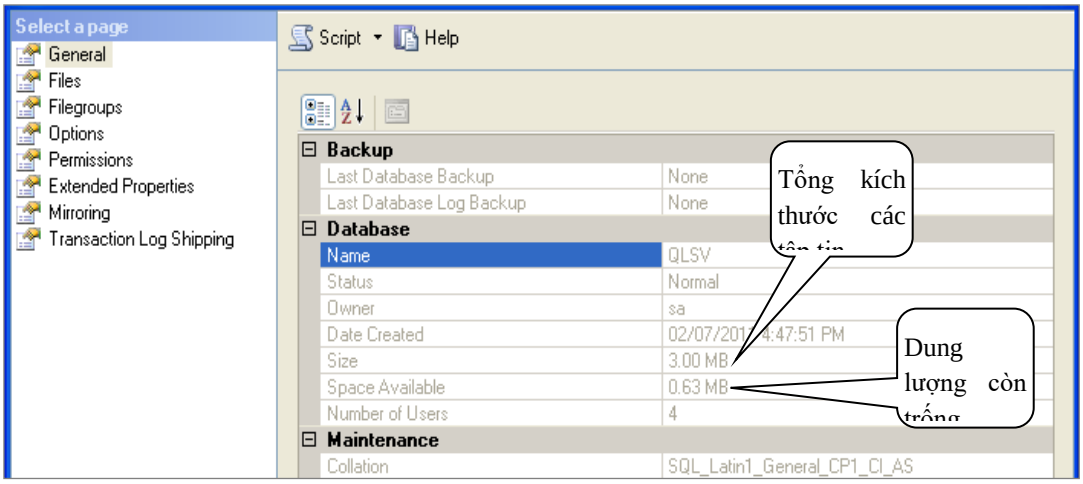
Sau khi câu lệnh trên được thực thi file log có tên là DB_SINHVIEN_log.ldf sẽ được tạo ra, được lưu trữ trong ổ đĩa D:\ và có kích thước ban đầu bằng 5MB.

2.1.2.3. Quản lý các tập tin trong cơ sở dữ liệu

Sau khi tạo ra cơ sở dữ liệu và đưa vào hoạt động thì khối lượng dữ liệu được lưu trữ lại ngày càng lớn nên việc theo dõi và quản lý dung lượng các tập tin trong cơ sở dữ liệu là rất cần thiết. Điều này sẽ giúp cho người quản trị cơ sở dữ liệu cũng như ban quản lý có những điều chỉnh hợp lý trên cơ sở dữ liệu.

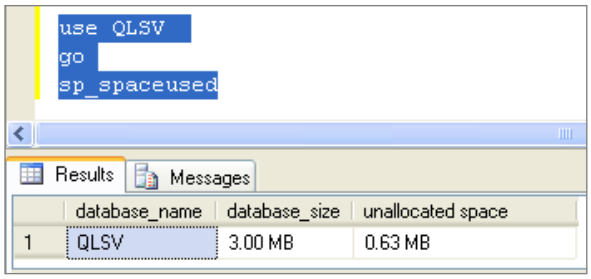
2.1.2.4. Xem kích thước cơ sở dữ liệu

Để theo dõi kích thước tập tin dữ liệu: Click chuột phải vào cơ sở dữ liệu cần theo dõi kích thước → Chọn Properties → Chọn General → tại mục Database, *Size* cho biết tổng kích thước của database, *Space Available* cho biết kích thước còn trống trong database.



Hình 2.8. Màn hình Properties của một database – Tùy chọn General

Ngoài ra, ta có thể sử dụng thủ tục hệ thống `sp_spaceused` để xem dung lượng:



Hình 2.9. Sử dụng cú pháp sp_spaceused và kết quả

Trong đó:

database_size: kích thước của database

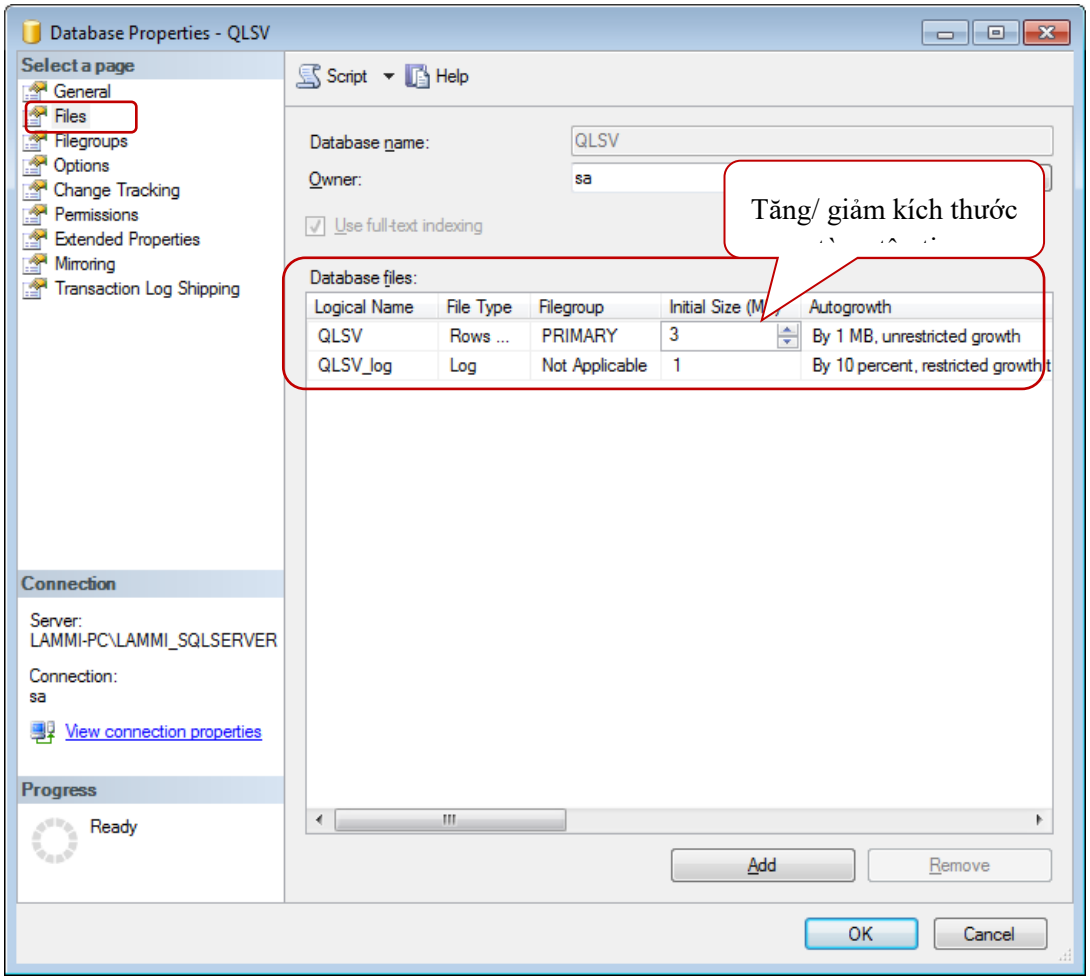
unallocated space: kích thước chưa phân bổ (dung lượng còn trống).

2.1.2.5. Thay đổi kích thước database

Khi kích thước trống của cơ sở dữ liệu không còn nhiều hoặc ngược lại kích thước trống của tập tin lớn hơn nhiều so với nhu cầu sử dụng thực tế sẽ dẫn đến lãng phí bộ nhớ lưu trữ, người quản trị cơ sở dữ liệu nghĩ ngay đến việc cấp thêm hoặc giảm bớt vùng nhớ cho cơ sở dữ liệu đó. Cụ thể là tăng/giảm kích thước cho các tập tin dữ liệu và tập tin log. Để thực hiện điều này, ta làm như sau:

Cách 1:

Click chuột phải vào cơ sở dữ liệu cần tăng/giảm kích thước → Chọn Properties → Chọn mục Files trên cây thư mục bên trái → Thực hiện tăng/giảm kích thước của các tập tin cần thiết trong vùng Database files



Hình 2.10. Màn hình Properties của một database – Tùy chọn File

Cách 2:

- **Tăng kích thước tập tin:**

```
Alter database <tên database>
Modify file( name= 'tên file', size=<số nguyên>MB)
```

Ví dụ 3: Tăng kích thước của tập tin có tên 'DB_PRIMARY' trong cơ sở dữ liệu DB_NHANVIEN lên thành 4MB (kích thước của tập tin này hiện tại là 3MB).

```
Alter database DB_NHANVIEN
Modify file(name='DB_PRIMARY', SIZE=4MB)
```

✍ **Lưu ý:** Kích thước chỉ định tăng của tập tin phải lớn hơn kích thước hiện tại của tập tin đó trong cơ sở dữ liệu.

- **Thu nhỏ kích thước tập tin:**

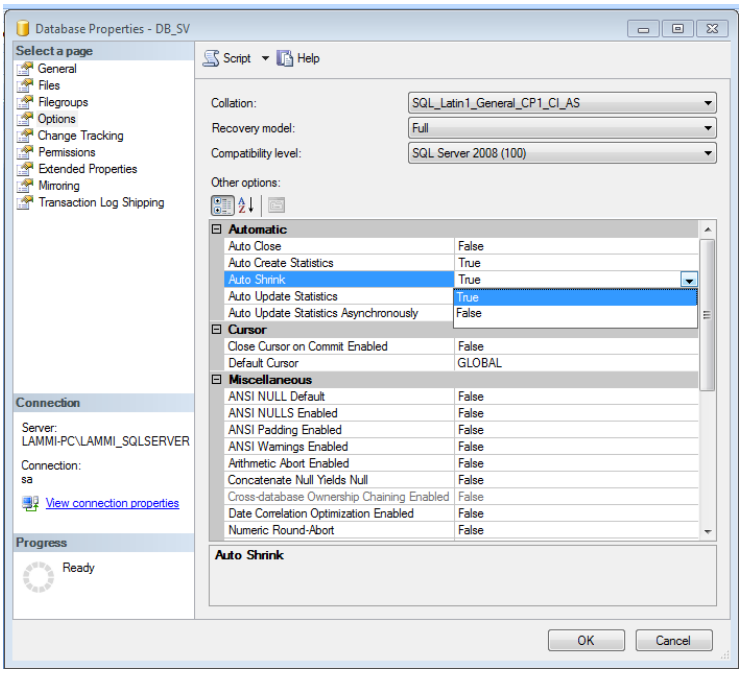
```
DBCC ShrinkFile
 (  <tên file>[, <kích thước file chỉ định(MB)>]
 )
```

Ví dụ 4: Thu nhỏ kích thước tập tin có tên là DB_PRIMARY thành 3 MB:

```
DBCC ShrinkFile
(
    DB_PRIMARY, 3
)
```

- Trường hợp giảm kích database bằng cách dùng chức năng *Auto Shrink* như sau:

Click chuột phải vào cơ sở dữ liệu cần giảm kích thước → Chọn Properties → Chọn mục Option → Auto Shrink: **True**



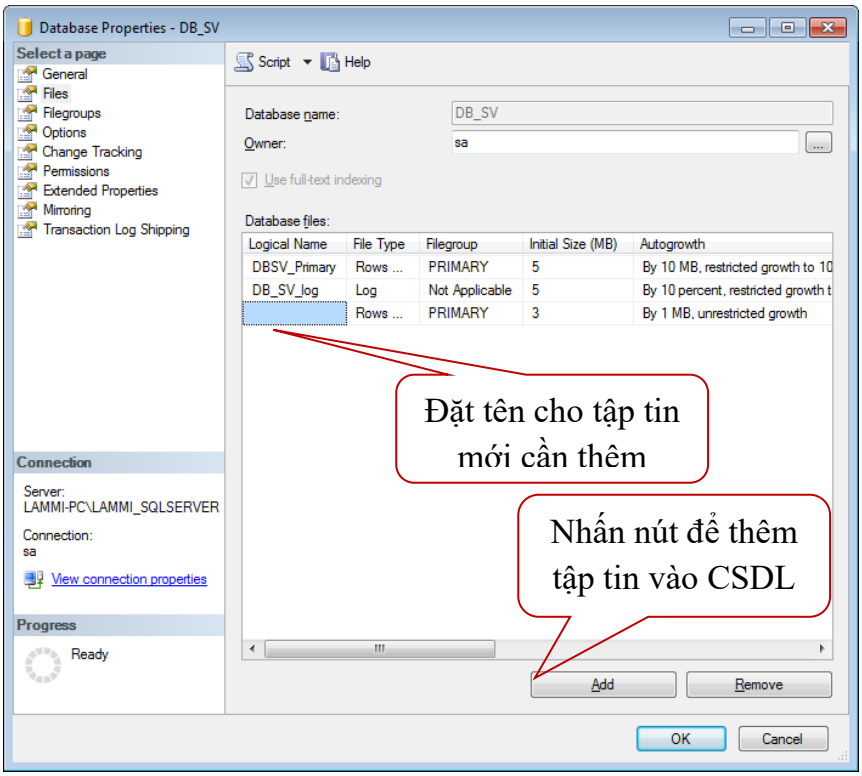
Hình 2.11. Màn hình Properties của một database – Tùy chọn Option

✍ **Lưu ý:** Khi thiết lập giá trị của Auto Shrink là True, hệ thống sẽ kiểm tra định kỳ tổng kích thước của tất cả các tập tin trong cơ sở dữ liệu và so sánh nó với khối lượng dữ liệu hiện đang lưu trữ. Nếu có hơn 25% không gian trống thì hệ thống sẽ tiến hành thu nhỏ kích thước tập tin để chỉ còn lại 25% không gian trống. Mặc định của tùy chọn này có giá trị là *False*.

2.1.2.6. Thêm mới tập tin trong cơ sở dữ liệu

Ngoài việc tăng kích thước cơ sở dữ liệu bằng cách tăng kích thước tập tin như trình bày trong phần trên, ta có thể thực hiện tăng kích thước cơ sở dữ liệu bằng cách thêm mới các tập tin.

Từ cửa sổ Database Properties nhấn nút Add để thêm tập tin mới và chỉ định các thông tin cần thiết.



Hình 2.12. Màn hình Properties của một database – Tùy chọn File

✍ **Lưu ý:** Tập tin mới thêm vào thuộc loại tập tin phụ (*secondary data file*), có phần mở rộng là *.ndf*.

Ngoài ra, có thể sử dụng lệnh T-SQL để thêm mới tập tin theo cú pháp sau:

Cú pháp:

- **Thêm mới một nhóm file**:

```
Alter database <tên database>
Add filegroup <tên nhóm file>
```

- **Thêm file vào nhóm**:

```
Alter database <tên database>
Add file | Add log file
(    name= 'data_file_logic',
     Filename= 'vị trí vật lý lưu trữ trên đĩa',
     Size=<dung lượng>,
     Maxsize=<dung lượng tối đa>,
     Filegrowth=<dung lượng tăng>
) to filegroup <tên filegroup>
```

Ví dụ 5: Trên cơ sở dữ liệu MyDB, thực hiện thêm file có tên là DB_Second2 vào nhóm có tên là nhom_1 (nhóm nhom_1 chưa được tạo), nơi lưu trữ vật lý là D:\DB_Second2.ndf.

- **Thực hiện tạo nhóm có tên là nhom_1:**

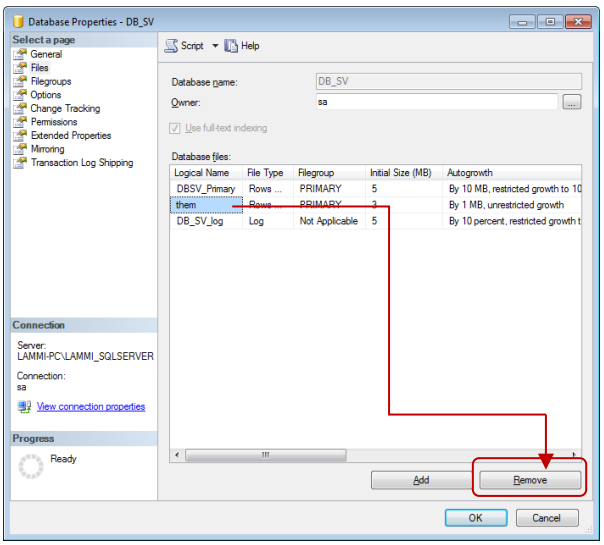
```
Alter database MyDB
Add filegroup nhom_1
```

- **Thêm file vào nhóm vừa tạo**

```
Alter database MyDB
Add file
(
    name= DB_Second2,
    Filename= 'D:\DB_Second2.ndf',
    Size=2MB,
    Maxsize=5MB,
    Filegrowth=1MB
)
to filegroup nhom_1
```

2.1.2.7. Hủy tập tin

Từ cửa sổ Database Properties chọn tập tin cần hủy và nhấn nút Remove để hủy bỏ tập tin.



Hình 2.13. Màn hình Properties của một database – Tùy chọn File

Hay có thể dùng lệnh T-SQL theo cấu trúc như sau:

```
Alter database <tên database>
Remove file <tên file>
```

Ví dụ 6: Hủy tập tin có tên là DB_Second2 trong cơ sở dữ liệu MyDB:

```
Alter database MyDB
Remove file DB_Second2
```

✍ **Lưu ý:** Hệ thống SQL Server chỉ cho phép hủy những tập tin trống, nghĩa là các tập tin này không chứa dữ liệu mới có thể hủy được. Sử dụng DBCC ShrinkFile với tùy chọn **EmptyFile** để làm trống một tập tin trước khi tiến hành hủy nó.

```
DBCC ShrinkFile
(
    DB_Second2,  EmptyFile
)
ALTER  DATABASE MyDB
Remove FILE DB_Second2;
```

2.2. Bảng (Table)

2.2.1. Khái niệm

Bảng là đối tượng lưu trữ dữ liệu chính trong SQL Server, việc tổ chức các bảng dựa vào mô hình cơ sở dữ liệu quan hệ, và được chuẩn hoá dựa vào các dạng chuẩn để dữ liệu được lưu trữ và sử dụng một cách hợp lý, nhất quán.

2.2.2. Xây dựng cấu trúc bảng

2.2.2.1. Tạo cấu trúc bảng bằng công cụ trên SQL Server Management Studio

Sau khi đã tạo xong CSDL ta tiến hành tạo các bảng để lưu trữ dữ liệu (Lưu ý: các bảng này đã được đưa ra trong mô hình quan hệ đã được thiết kế trước đó)

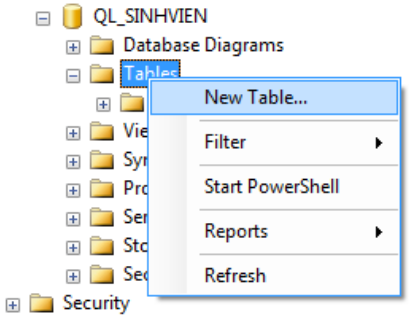
Giả sử rằng trong lược đồ quan hệ đã được thiết kế trước đó có 2 lược đồ như sau:

LOP (MALOP, TENLOP, SISO)

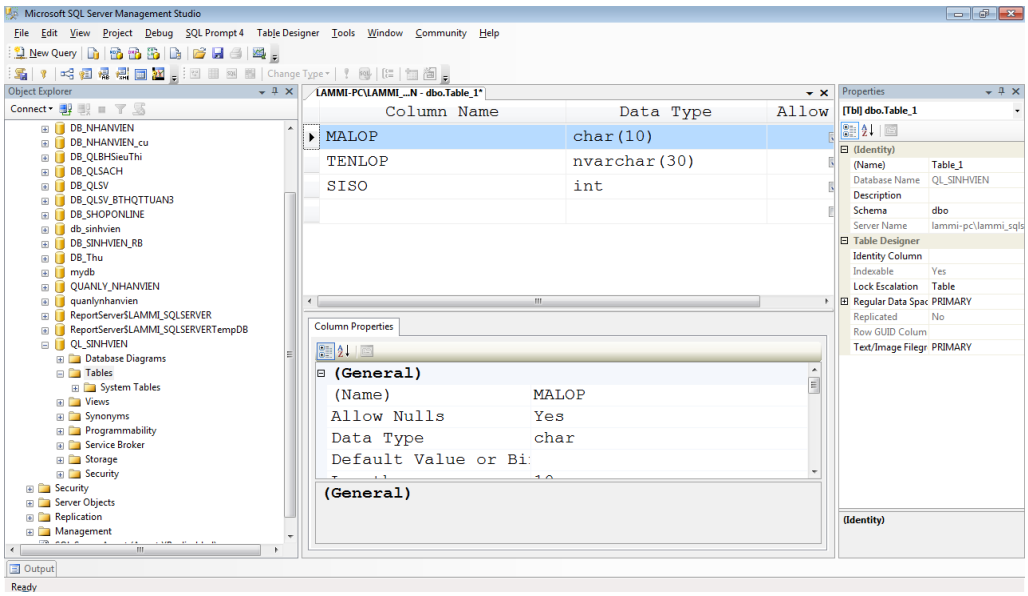
SINHVIEN (MASV, HOTEN, NGSINH, PHAI, QUEQUAN, MALOP)

Bây giờ ta tiến hành tạo 2 bảng tương ứng với 2 lược đồ này, thực hiện như sau:

Mở rộng danh mục cơ sở dữ liệu QL_SINHVIEN → click phải chuột vào mục Tables → chọn *New Table*



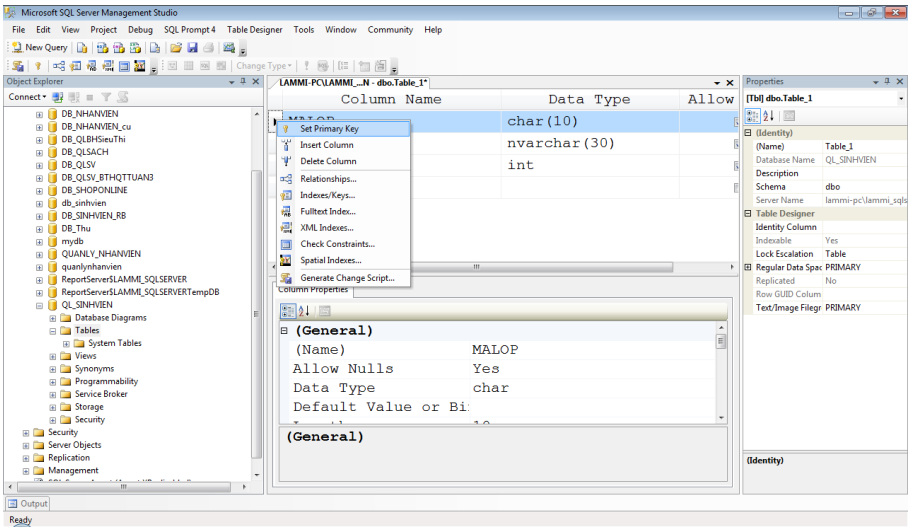
Khi chọn *New Table* sẽ xuất hiện bên phải màn hình bên dưới.



Hình 2.14. Màn hình Design một table

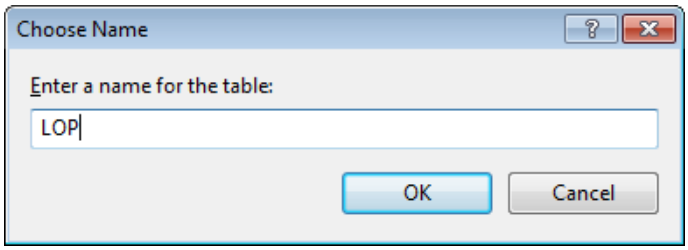
Nhập thông tin từng cột của bảng cần tạo vào các vị trí *Column Name* (Tên cột), *Data Type* (Kiểu dữ liệu), Allow Nulls (Cho phép rỗng hay không). Nhấn Enter để nhập cột kế tiếp.

Có thể tạo khóa chính bằng cách click phải chuột vào cột cần định nghĩa khóa chính → chọn *Primary Key* hoặc chọn công cụ khóa trên thanh công cụ.



Hình 2.15. Màn hình Design một table – Set Primary key

Và cuối cùng lưu bảng lại bằng cách nhấn công cụ *Save* trên thanh công cụ.



Hình 2.16. Màn hình lưu table

Tương tự cho việc tạo bảng **SINHVIEN**.

Tạo khóa ngoại (Foreign key) trong cửa sổ thiết kế table. Foreign Key được dùng để liên kết các bảng lại với nhau.

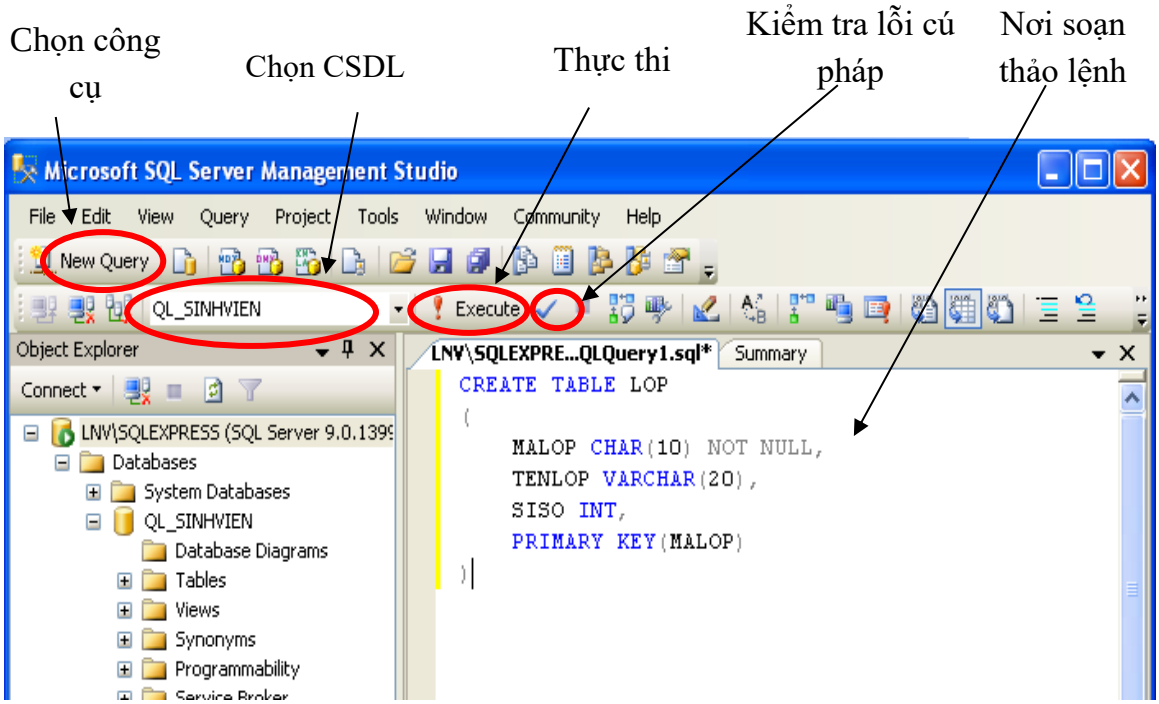
Ví dụ 7: Trong 2 table sau trường MaLOP trong SINHVIEN làm Foreign key

LOP (MALOP, TENLOP, SISO)

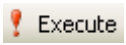
SINHVIEN (MASV, HOTEN, NGSINH, PHAI, QUEQUAN, MALOP)

2.2.2.2. Tạo cấu trúc bảng bằng lệnh T – SQL

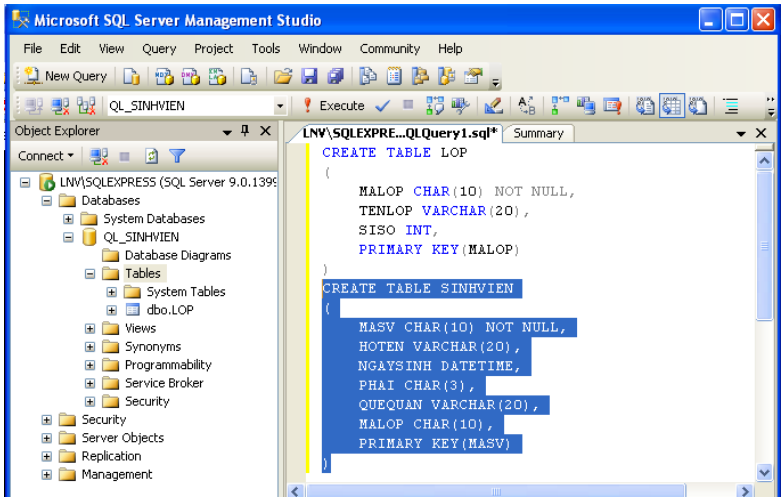
Mở cửa sổ soạn thảo lệnh bằng cách chọn công cụ "*New Query*" trên thanh công cụ sau đó chọn CSDL muốn tạo bảng trên thanh công cụ và nhập lệnh tạo bảng vào cửa sổ soạn thảo lệnh như hình dưới đây:



Hình 2.17. Giao diện viết lệnh

Sau khi nhập lệnh xong ta kiểm tra lỗi cú pháp bằng công cụ  và thực thi bằng công cụ  trên thanh công cụ. Khi thực thi lệnh tạo bảng trên nếu không có lỗi xảy ra thì một bảng LOP sẽ được tạo ra trong cơ sở dữ liệu đã chỉ định tạo bảng.

Để tạo tiếp bảng SINHVIEN ta không mở cửa sổ lệnh mới mà nhập tiếp lệnh tạo bảng SINHVIEN bên dưới lệnh tạo bảng LOP trước đó rồi chọn (tô đen) khối lệnh vừa nhập, kiểm tra lỗi cú pháp và thực thi tương tự.(✍ **Lưu ý**: có thể thực thi lệnh mà không cần kiểm tra lỗi cú pháp)



Hình 2.18. Giao diện viết lệnh

✍ **Lưu ý:** khi thiết kế một table:

– Phải nắm vững về các loại Data type.

- Xác định khóa chính chính xác.
- Tránh dùng cột có chứa NULL và nên luôn có giá trị Default cho các cột.
- Phải biết rõ quan hệ (Relationship) giữa các table.

❖ **Nhắc lại các loại ràng buộc toàn vẹn trên các cột trong một table:**

- NOT NULL (không cho rỗng), NULL (cho rỗng)
- UNIQUE (kiểm tra tính duy nhất)
- DEFAULT (giá trị mặc định)
- PRIMARY KEY (khóa chính)
- FOREIGN KEY/ REFERENCES (khóa ngoại)
- CHECK (kiểm tra miền giá trị)

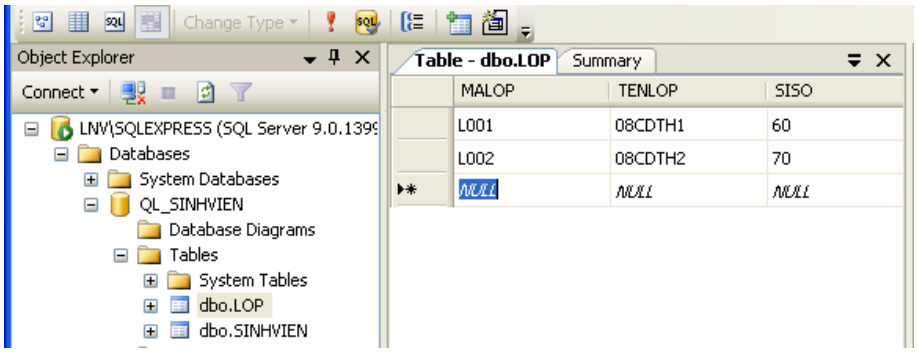
(Tham khảo Bài giảng thực hành Cơ sở dữ liệu, khoa CNTT)

2.2.3. Nhập liệu vào bảng

Sau khi tạo bảng và lược đồ Diagram (xem trong phần 2.3) thì bước tiếp theo là nhập dữ liệu vào bảng (✍ **Lưu ý**: khi nhập liệu vào bảng phải nhập theo đúng thứ tự là bảng có liên kết một nhập trước, bảng có liên kết nhiều nhập sau. Có 3 hình thức nhập liệu vào bảng như sau:

2.2.3.1. Nhập trực tiếp

Trong cơ sở dữ liệu QL_SINHVIEN mở rộng danh mục Tables → nhấp chuột phải vào bảng LOP → chọn Edit Top 200 Rows → nhập liệu vào bảng.



Hình 2.19. Giao diện nhập dữ liệu vào bảng

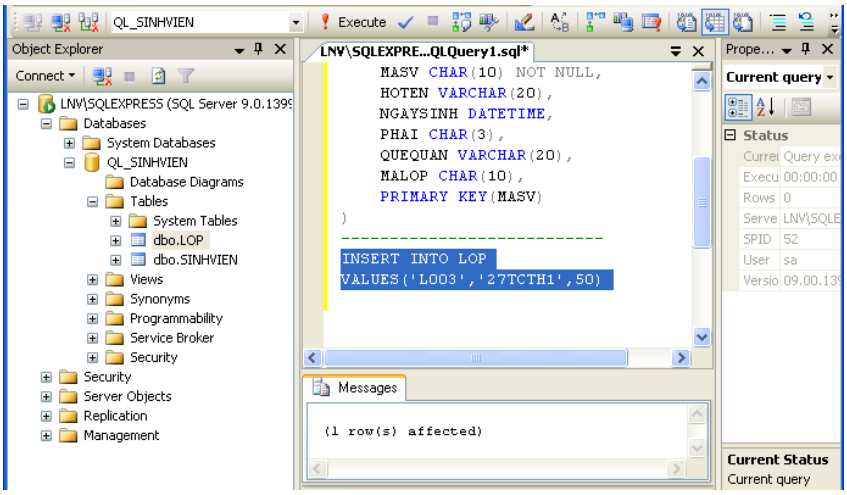
2.2.3.2. Nhập bằng lệnh

Việc nhập liệu trực tiếp vào bảng không hỗ trợ người sử dụng trong trường hợp thao tác trên các trình ứng dụng khác có kết nối với SQL Server, để giải quyết vấn đề này ta dùng cách nhập liệu bằng lệnh.

Cú pháp:

```
INSERT INTO <tên bảng> [tên các cột]
VALUES(các giá trị tương ứng)
```

Minh hoạ cho cách nhập liệu này bằng cách gõ lệnh nhập liệu vào bảng LOP trên cửa sổ soạn thảo lệnh:



Hình 2.20. Giao diện viết lệnh nhập dữ liệu vào bảng

Sau khi gõ xong lệnh, chọn khối lệnh vừa gõ rồi kiểm tra lỗi cú pháp và thực thi. Nếu không có lỗi xảy ra thì một dòng dữ liệu được thêm vào bảng LOP.

2.2.3.3. Nhập từ một bảng khác

Dữ liệu cần nhập vào bảng có thể được lấy từ một bảng khác bằng cách thực thi câu lệnh Select.

Giả sử trong cơ sở dữ liệu QL_SINHVIEN (được tạo ra ở trên) ta tạo thêm một bảng dữ liệu có tên SV_08CDTH1 gồm 2 cột là MASV, HOTEN. Vấn đề đặt ra: Trích ra những sinh viên học lớp 08CDTH1 (TENLOP = '08CDTH1') từ bảng SINHVIEN và chèn vào bảng SV_08CDTH1.

Thực hiện điều này bằng cách thực thi câu lệnh như sau:

```
INSERT INTO SV_08CDTH1
    SELECT MASV, HOTEN
    FROM SINHVIEN, LOP
    WHERE SINHVIEN.MALOP=LOP.MALOP AND
          TENLOP='08CDTH1'
```

2.2.3.4. Một số lưu ý

Khi nhập dữ liệu vào bảng cần lưu ý một số trường hợp đặc biệt sau:

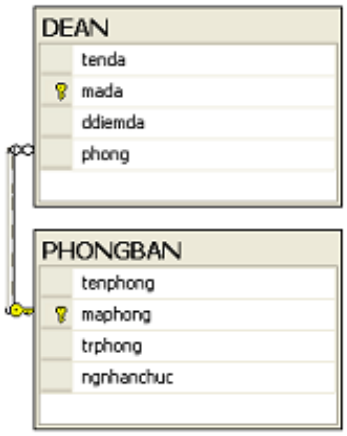
- **Thêm ký tự N trước chuỗi Unicode (Chuỗi nhập có dấu tiếng Việt)**

Ví dụ 8: Thêm vào bảng NHANVIEN(MANV, TENNV, GIOITINH) một bộ giá trị ('NV01', 'Nguyễn Văn Trường', 'Nam')

```
Insert into NHANVIEN
Values ('NV01', N'Nguyễn Văn Trường', 'Nam')
```

- **Nhập dữ liệu cho các bảng khi đã có ràng buộc khóa ngoại**

Ví dụ 9: Thứ tự nhập dữ liệu cho các bảng trong liên kết sau (liên kết 1–n, phong trong DEAN là khóa ngoại liên kết với maphong làm khóa chính trong bảng PHONG):



Hình 2.21. Lược đồ Diagram kết nối giữa DEAN và PHONGBAN

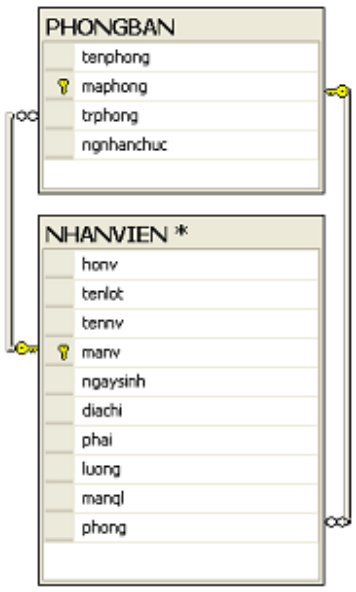
Cách 1:

- Bước 1: Nhập PHONGBAN
- Bước 2: Nhập DEAN

Cách 2:

- Bước 1: Nhập DEAN, nhập giá trị cho cột phong NULL
- Bước 2: Nhập PHONGBAN
- Bước 3: Cập nhật dữ liệu cho cột phong trong DEAN

Ví dụ 10: Thứ tự nhập dữ liệu cho các bảng trong liên kết sau (liên kết 1–n thứ 1: trphong trong PHONGBAN là khóa ngoại liên kết với manv làm khóa chính trong bảng NHANVIEN, liên kết 1–n thứ 2: phong trong NHANVIEN là khóa ngoại liên kết với maphong làm khóa chính trong bảng PHONGBAN):



Hình 2.22. Lược đồ Diagram kết nối giữa **PHONGBAN và NHANVIEN**

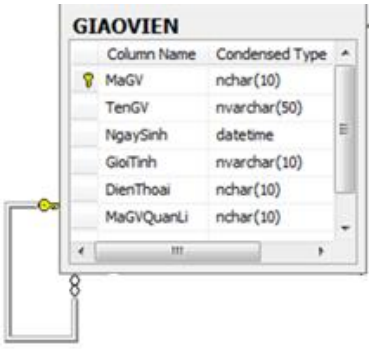
Cách 1:

- Bước 1: Nhập NHANVIEN, nhập giá trị cho cột phong NULL
- Bước 2: Nhập PHONGBAN
- Bước 3: Cập nhật thuộc tính phong trong NHANVIEN

Cách 2:

- Bước 1: Nhập PHONGBAN, nhập giá trị cho cột trphong NULL
- Bước 2: Nhập NHANVIEN
- Bước 3: Cập nhật thuộc tính trphong trong PHONGBAN

Ví dụ 11: Thứ tự nhập dữ liệu cho bảng sau (liên kết 1–n: MaGVQuanLi trong GIAOVIEN là khóa ngoại liên kết với MaGV làm khóa chính trong bảng GIAOVIEN):



Hình 2.23. Lược đồ quan hệ NHANVIEN

Một thể hiện của bảng GIAOVIEN:

MaGV	TenGV	NgaySinh	GioiTinh	DienThoai	MaGVQuanLi
GV00001	Nguyễn Văn An	1981-01-02 00:...	Nam	NULL	GV00002
GV00002	Nguyễn Thị Như Lan	1984-12-02 00:...	Nữ	NULL	GV00005
GV00003	Trần Minh Anh	1986-03-23 00:...	Nam	0909123999	GV00002
GV00004	Trương Tường Vi	1988-02-01 00:...	Nữ	0998990909	GV00008
GV00005	Hà Anh Tuấn	1986-12-03 00:...	Nam	0909909000	GV00008
GV00006	Trần Anh Dũng	1979-04-04 00:...	Nam	NULL	GV00010
GV00007	Trần Duy Tân	1978-01-04 00:...	Nam	NULL	GV00002
GV00008	Nguyễn Thị Linh	1979-07-08 00:...	Nữ	0938079700	GV00009
GV00009	Trần Thị Kiều	1977-01-03 00:...	Nữ	NULL	NULL
GV00010	Trần Phương Loan	1978-04-30 00:...	Nữ	NULL	NULL

Hình 2.24. Thể hiện của quan hệ GIAOVIEN

Cách 1:

- Bước 1: Những giáo viên có MaGVQuanLi NULL nhập trước.
- Bước 2: Sau đó nhập những giáo viên mà đã nhập thông tin người quản lý của giáo viên đó.

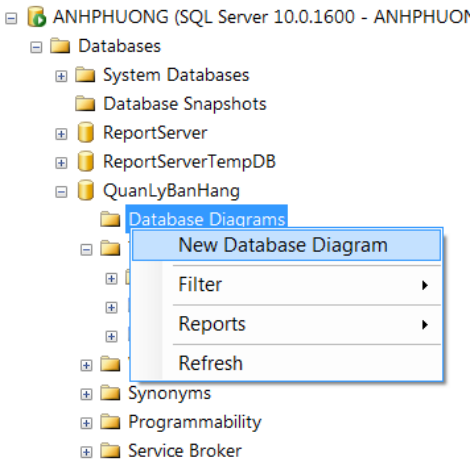
Cách 2:

- Bước 1: Nhập GIAOVIEN đặt MaGVQuanLi NULL.
- Bước 2: Cập nhật cột MaGVQuanLi trong bảng GIAOVIEN.

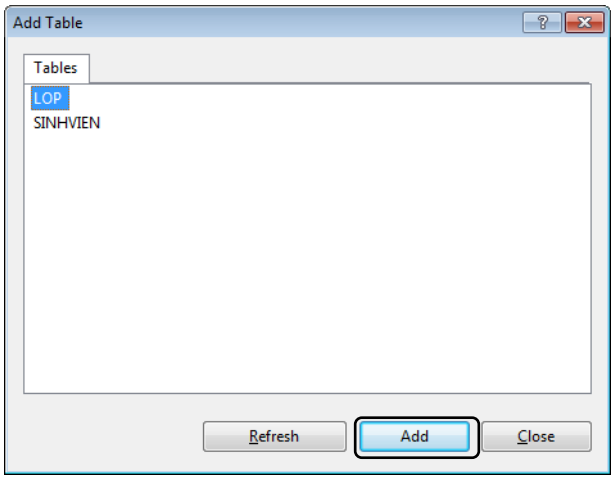
2.2.4. Lược đồ Diagram

Đối tượng *Diagram* là một lược đồ thể hiện sự liên kết các bảng trong cơ sở dữ liệu với nhau. Trên lược đồ *Diagram* chỉ thể hiện 2 loại mối liên kết là 1–n và 1–1.

Tạo lược đồ Diagram bằng cách: Mở rộng danh mục cơ sở dữ liệu QL_SINHVIEN → nhấp phải chuột vào *Database Diagrams* → chọn *New Database Diagram*

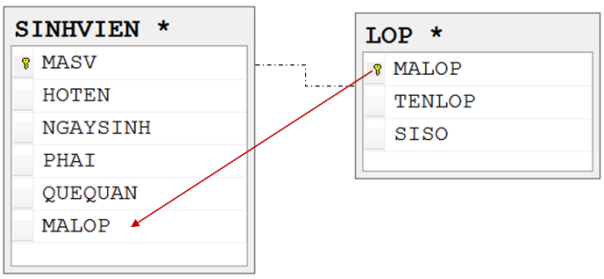


Trên cửa sổ *Add Table*, chọn các bảng cần đưa vào lược đồ → nhấn nút *Add*.



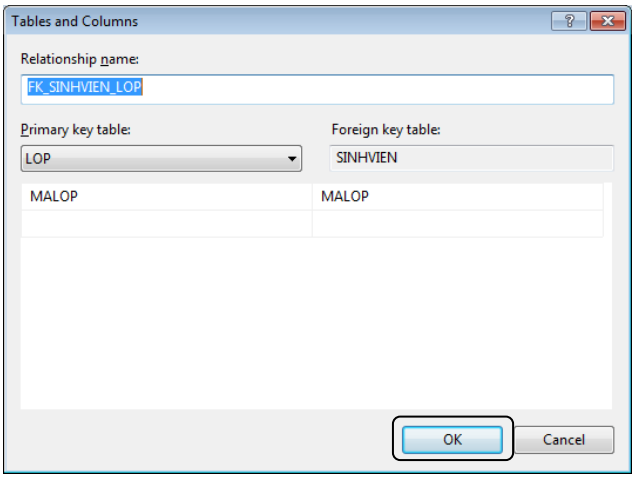
Hình 2.25. Cửa sổ Add Table

Tạo mối liên kết bằng cách nhấp giữ chuột trái vào cột MALOP trên bảng SINHVIEN rồi kéo sang bảng LOP và thả chuột ra



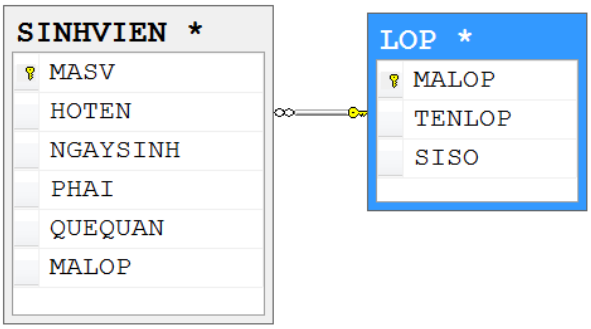
Hình 2.26. Thực hiện kết nối giữa SINHVIEN và LOP

Sau khi kéo mối quan hệ cho 2 table sẽ xuất hiện hộp thoại như hình. Trên cửa sổ *Table and Columns*, ta đặt tên ràng buộc và kiểm tra lại thông tin thuộc tính khóa ngoại và khóa chính trên hai bảng phải là MALOP. Nếu không phải MALOP thì click chuột vào tên thuộc tính và chọn lại cho đúng thuộc tính MALOP. Sau khi kiểm tra xong, nhấn OK để hoàn tất.



Hình 2.27. Giao diện Tables and Columns

Click chọn OK, khi đó một mối liên kết 1 – n sẽ được tạo ra.



Hình 2.28. Lược đồ Diagram kết nối giữa SINHVIEN và LOP

Trong cùng một cách, bạn có thể tạo mối quan hệ khác. Khi bạn đã hoàn tất, bạn có thể lưu và đóng diagram.

✍ **Lưu ý**: Để tạo mối liên kết 1–1 giữa hai bảng, chẳng hạn như bảng TRPHONG và bảng PHONG với cấu trúc:

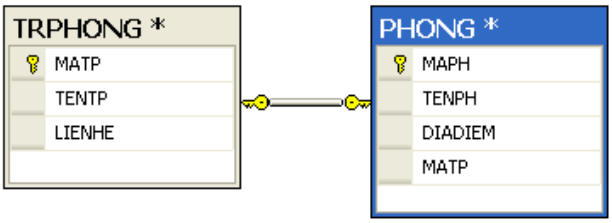
TRPHONG(MATP, TENTP, LIENHE)

PHONG(MAPH, TENPH, DIADIEM, MATP)

Thuộc tính MATP trong bảng PHONG có vai trò là khóa ngoại tạo liên kết giữa hai bảng nên để tạo liên kết 1-1 cần thiết lập ràng buộc là *UNIQUE* trên thuộc tính này.

```
CREATE TABLE PHONG
(
        MAPH CHAR(10) NOT NULL,
        TENPH VARCHAR(30),
        DIADIEM VARCHAR(50),
        MATP NCHAR(10) UNIQUE,
        PRIMARY KEY(MAPH)
)
```

Sau khi tạo xong hai bảng PHONG và TRPHONG, tiến hành tạo liên kết tương tự như cách thức đã trình bày ở trên, khi đó một mối liên kết 1 – 1 sẽ được tạo ra.



Hình 2.29. Lược đồ Diagram kết nối giữa TRPHONG và PHONG

2.3. Bảng ảo (View)

Bảng ảo là một đối tượng được tạo ra từ các bảng cơ sở. Mục đích của việc tạo *View* là để tạo ra các bảng trung gian nhằm đơn giản hóa các câu truy vấn phức tạp, che dấu dữ liệu ngoài ra còn dùng với mục đích bảo mật.

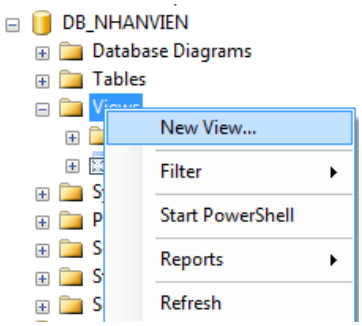
Bảng ảo cũng là một quan hệ, tuy nhiên:

- Không chứa dữ liệu
- Được định nghĩa từ bảng khác

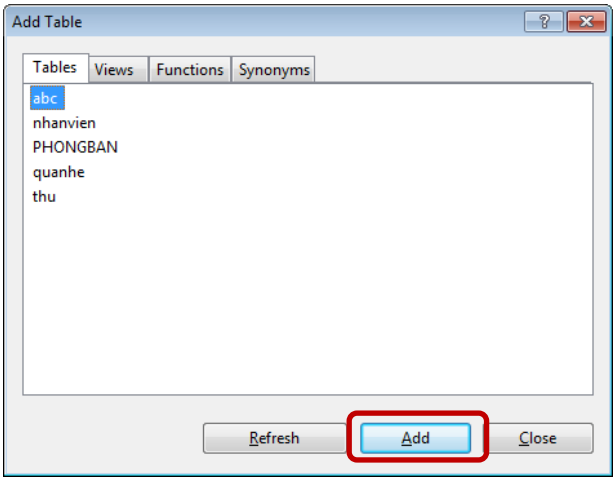
2.3.1. Tạo view bằng công cụ

Khởi động SQL Server Management Studio

- Kết nối với Server
- Chọn cơ sở dữ liệu cần làm việc
- Click phải vào mục *Views* → chọn *New View*

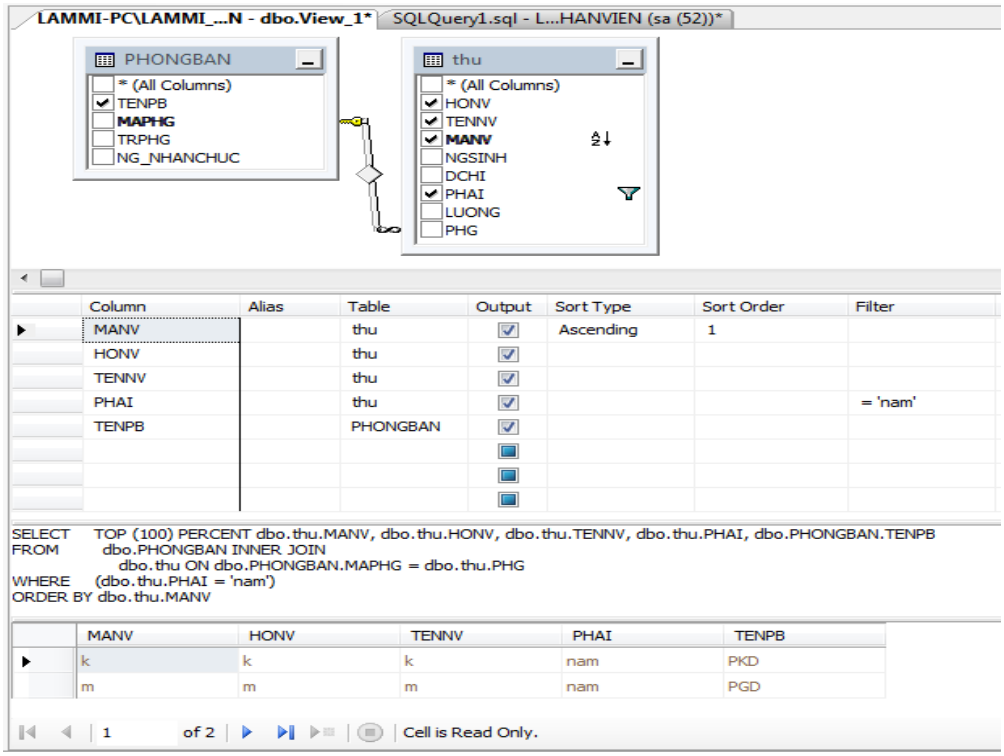


- Trong hộp thoại *Add Table* → chọn các bảng cần thiết liên quan đến *View* cần tạo → *Add*



Hình 2.30. Giao diện Add Table

- Chọn tên các cột cần hiển thị (*Column*), tên bí danh (*Alias*), điều kiện lọc (*filter*)

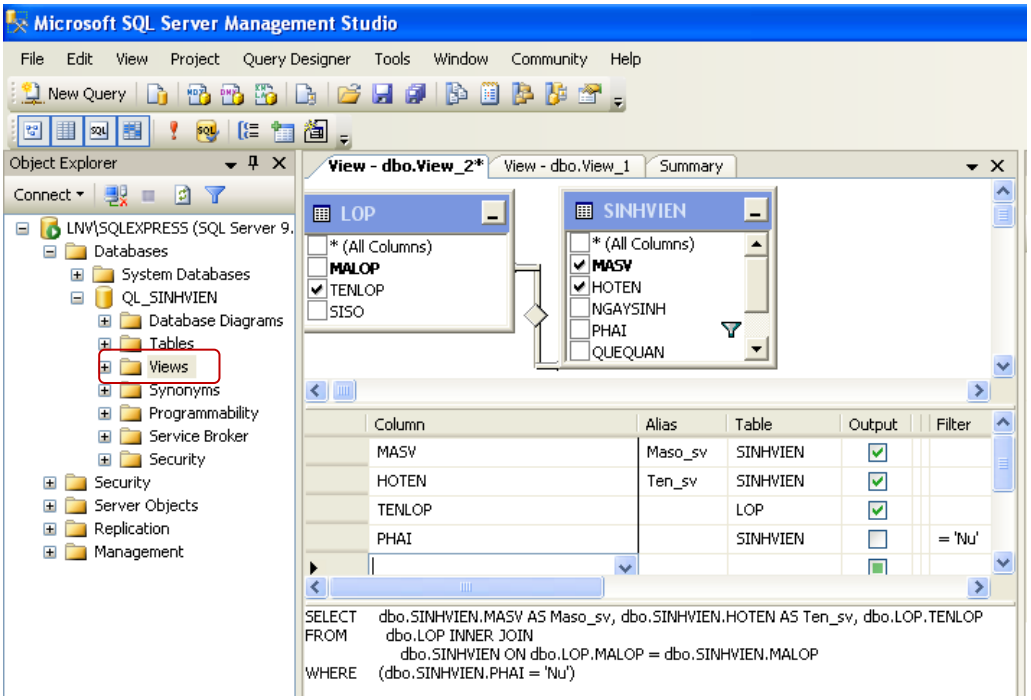


Hình 2.31. Giao diện thiết kế View

– Chọn *Execute SQL* để xem kết quả

– Ctrl + S để lưu View (nếu cần)

Ví dụ 12: Tạo một View có tên là V1 gồm 3 cột là MASV, HOTEN, TENLOP từ 2 bảng cơ sở LOP, SINHVIEN với điều kiện là các sinh viên phải là '*Nu*'.



Hình 2.32. Giao diện thiết kế View

2.3.2. Tạo view bằng lệnh T-SQL

Cú pháp:

```
CREATE VIEW <tên_view>
AS
  Câu lệnh SELECT
```

Mở cửa sổ soạn thảo lệnh, gõ vào đoạn lệnh tạo view rồi thực thi bằng cách chọn khối lệnh vừa gõ và nhấp chọn công cụ  (Execute SQL) trên thanh công cụ. Khi đó một view sẽ được tạo ra.

Ví dụ 13: Tạo View chứa thông tin các nhân viên ở phòng 'DH' trong bảng NhanVien:

MANV	HOTEN	NTNS	PHAI	MA_NQL	MaPH	LUONG
001	Vuong Ngoc Quyen	22/10/1957	Nu		QL	3.000.000
002	Nguyen Thanh Tung	09/01/1955	Nam	001	NC	2.500.000
003	Le Thi Nhan	18/12/1960	Nu	001	DH	2.500.000
004	Dinh Ba Tien	09/01/1968	Nam	002	NC	2.200.000
005	Bui Thuy Vu	19/07/1972	Nam	003	DH	2.200.000

Hình 2.33. Thể hiện của quan hệ NHANVIEN

```
Create View View_1
As
    Select    *
    From      NhanVien
    Where     MaPH = 'DH'
```

✍ **Lưu ý**: Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức là không phải là một cột trong bảng dữ liệu cơ sở) mà cột đó không được đặt tên thì sẽ là lỗi.

Ví dụ 14: Ví dụ sau khi thực thi sẽ báo lỗi

```
Create View View_2
As
  Select      MaPH, TenPH, Count(*)
  From    NhanVien    nv, PhongBan     pb
  Where   nv. MaPH = pb.MaPH
  Group by    MaPH, TenPH
```

Lỗi vì cột chưa được đặt tên

Câu lệnh đúng phải là:

```
Create View View_2
As
  Select     MaPH, TenPH, Count(*) As  SLNV
  From       NhanVien   nv, PhongBan    pb
  Where      nv. MaPH = pb.MaPH
  Group by   MaPH, TenPH
```

2.3.3. Sửa View bằng lệnh T – SQL

Cú pháp:

```
ALTER  VIEW  ⟨Tên_View⟩[(danh_sách_cột)]
AS
  ⟨Câu_lệnh_Select⟩
```

2.3.4. Xóa View bằng lệnh T – SQL

Cú pháp:

```
DROP  VIEW  ⟨Tên_View⟩
```

2.4. Truy vấn dữ liệu

Tham khảo sách Giáo trình Cơ sở dữ liệu, khoa CNTT.

Kết chương

Cơ sở dữ liệu, bảng, diagram, view, ràng buộc khoá chính, khoá ngoại là các đối tượng cơ sở quan trọng trong quá trình xây dựng và quản lý dữ liệu. Hiểu được cấu trúc và kiểm soát được kích thước các tập tin dữ liệu là một công việc rất quan trọng trong quá trình vận hành và quản lý cơ sở dữ liệu.

Câu hỏi và bài tập chương 2

Câu 1: Khởi động SQL Server Management Studio, đăng nhập và tạo một cơ sở dữ liệu có tên là QUANLY_SINHVIEN

a/ Dùng lệnh T – SQL thực hiện tạo các bảng tương ứng với lược đồ cơ sở dữ liệu sau:

KHOA (MAKHOA, TENKHOA)

LOP (MALOP, TENLOP, SISODK, MAKHOA)

SINHVIEN (MASV, HOTEN, NGSINH, DCHI, GIOITINH, MALOP)

MONHOC (MAMH, TENMH, SOTIET)

KETQUA (MASV, MAMH, DIEM)

b/ Dùng lệnh T – SQL tạo ràng buộc khóa ngoại có trong cơ sở dữ liệu

c/ Tạo lược đồ Diagram

d/ Dùng lệnh INSERT INTO nhập liệu vào các bảng với dữ liệu như dưới đây:

KHOA

MAKHOA	TENKHOA
TH	TIN HOC
VL	VAT LY
DT	DIEN TU
TP	THUC PHAM
HH	HOA HOC
SH	SINH HOC

LOP

MALOP	TENLOP	SISODK	MAKHOA
L001	24THTH1	30	TH
L002	24THTH2	30	TH
L003	05CDTH	60	TH
L004	06CDTH1	60	TH
L005	06CDTH2	60	TH
L006	24THTP1	30	TP
L007	24THTP2	30	TP
L008	24THTP3	30	TP
L009	06CDDT	60	DT
L010	05CDVL	45	VL
L011	24THHH	60	HH
L012	06CDSH	60	SH

SINHVIEN

MASV	HOTEN	NGSINH	DCHI	GIOITINH	MALOP
SV01	Nguyen Thi Lan	12/03/1978	TPHCM	Nam	L001
SV02	Tran Thanh Tung	02/04/1980	VUNG TAU	Nam	L001
SV03	Truong Thi Hue	11/10/1984	DA NANG	Nu	L001
SV04	Le Van Khanh	04/07/1985	VUNG TAU	Nam	L002
SV05	Ngo Viet Nam	14/08/1986	DA NANG	Nam	L002
SV06	Tran Thi Lieu	15/07/1985	TPHCM	Nu	L003

SV07	Tran Thanh Nam	02/04/1988	DONG NAI	Nam	L004
SV08	Pham Hoai Phong	19/05/1979	TIEN GIANG	Nam	L004
SV09	Tran Thi To Anh	08/07/1981	TPHCM	Nu	L004
SV10	Do Thi Hanh	15/03/1982	DONG NAI	Nu	L004
SV11	Ha Thanh Tung	26/04/1983	VUNG TAU	Nam	L005
SV12	Tran Quoc Tuan	15/07/1982	TIEN GIANG	Nam	L006
SV13	Nguyen Minh Tri	10/10/1981	DONG THAP	Nam	L007
SV14	Tran Thanh Phuoc	17/12/1984	TPHCM	Nam	L008
SV15	Nguyen Phuoc Sang	18/02/1986	HA NOI	Nam	L008
SV16	Le Thi Anh	15/05/1987	HA NOI	Nu	L009
SV17	Truong Thi Nhan	12/03/1981	TPHCM	Nu	L010
SV18	Nguyen Kim Phung	14/06/1983	HA NOI	Nu	L011
SV19	Tran Thi Tham	11/02/1985	DONG NAI	Nu	L012
SV20	N Thi Thu Trang	17/05/1988	TPHCM	Nu	L005

MONHOC

MAMH	TENMH	SOTIET
M001	TOAN CAO CAP A1	60
M002	LICH SU DANG	45
M003	CHINH TRI	45
M004	CO SO DU LIEU	90
M005	SQL SERVER	60
M006	LAP TRINH C	60
M007	XU LY ANH	90
M008	TIN HOC CAN BAN	60
M009	MANG MAY TINH	60
M010	TOAN ROI RAC	45
M011	LAP TRINH Q.LY	90
M012	VISUAL BASIC	60

KETQUA

MASV	MAMH	DIEM
SV01	M001	8
SV01	M002	4
SV01	M003	6
SV02	M001	7
SV04	M001	5
SV04	M002	7

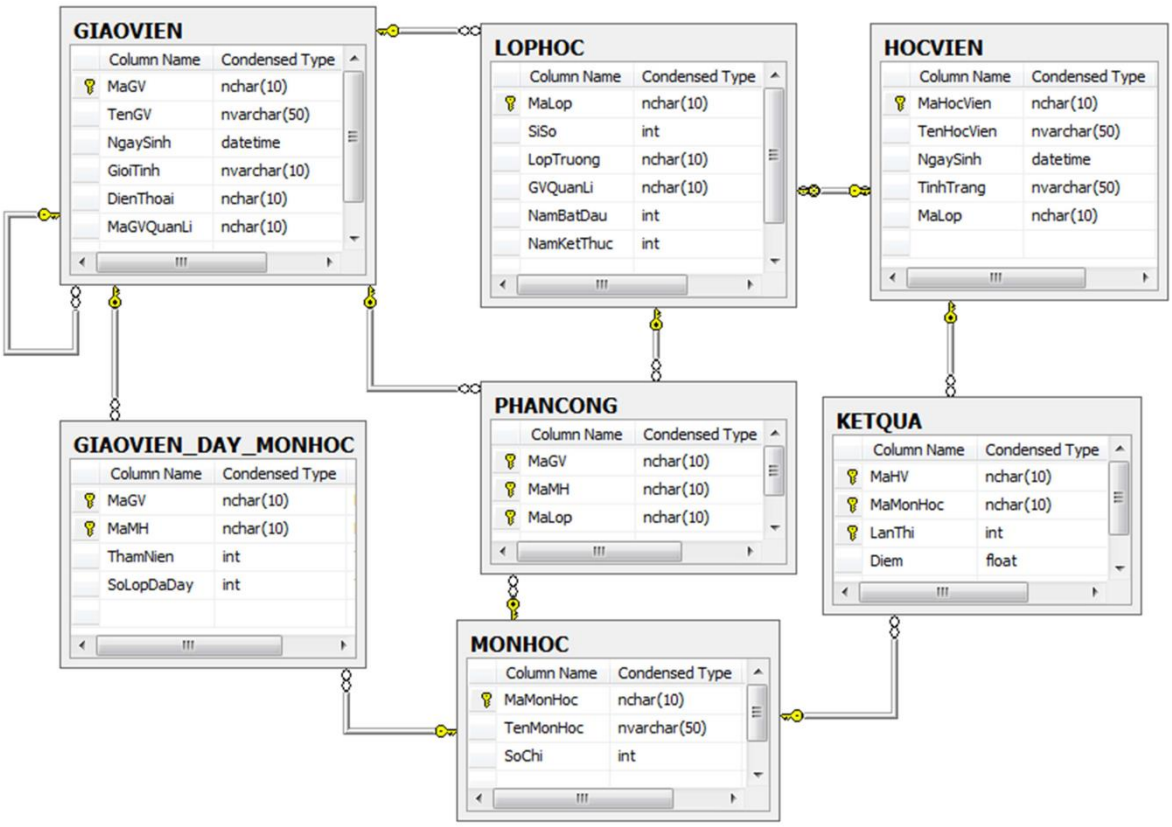
SV05	M003	9
SV06	M008	10
SV07	M005	6
SV07	M007	9
SV08	M004	7
SV09	M010	3
SV10	M001	8
SV10	M006	6
SV11	M004	5
SV12	M004	8
SV13	M007	8
SV14	M005	4
SV14	M006	7
SV15	M009	7
SV16	M002	10
SV17	M006	8
SV18	M007	9
SV19	M006	7
SV20	M001	5
SV20	M002	7
SV20	M004	6
SV20	M005	9
SV20	M009	10

Câu 2: Khởi động SQL Server Management Studio, đăng nhập và tạo một cơ sở dữ liệu quản lý học viên có tên là **DB_QLHV,** yêu cầu:

Một tập tin chính có tên logic là **DBQLHV_PRIMARY**, tên lưu trữ vật lý trên đĩa là **DBQLHV_PRIMARY.mdf**, kích thước ban đầu là: 5MB, tỉ lệ tăng trưởng là 10%, kích thước tối đa là 10MB.

Tập tin transaction log có tên logic là: **DBQLHV_LOG**, tên lưu vật lý lưu trữ trên đĩa là **DBQLHV_LOG.ldf**, kích thước ban đầu là: 3MB, tỉ lệ tăng trưởng là 15%, kích thước tối đa là 5MB.

Dùng lệnh T – SQL thực hiện tạo các bảng tương ứng với lược đồ cơ sở dữ liệu sau:



a/ Dùng lệnh T – SQL tạo ràng buộc khoá ngoại có trong cơ sở dữ liệu (tương ứng với các đường mũi tên tham chiếu)

b/ Tạo lược đồ Diagram

c/ Dùng lệnh INSERT INTO hoặc SSMS nhập liệu vào các bảng với dữ liệu như dưới đây:

HOCVIEN

Table - dbo.HOCVIEN | (local).QLHocVien - QLHV_Answer.sql | Summary

MaHocVien	TenHocVien	NgaySinh	TinhTrang	MaLop
HV000001	Nguyễn Thùy Linh	01/02/1990 12:...	buộc thôi học	LH000001
HV000002	Nguyễn Thị Kiều ...	20/12/1993 12:...	đang học	LH000001
HV000003	Nguyễn Xuân Thu	30/12/1994 12:...	đang học	LH000002
HV000004	Trần Trung Chính	12/03/1992 12:...	đang học	LH000003
HV000005	Trần Minh An	03/12/1991 12:...	đang học	LH000003
HV000006	Trương Mỹ Linh	12/12/1989 12:...	đã tốt nghiệp	LH000004
HV000007	Trần Hào	02/02/1989 12:...	đã tốt nghiệp	LH000004
HV000008	Nguyễn Huỳnh	03/03/1992 12:...	đang học	LH000004
HV000009	Nguyễn Xuân Tr...	13/03/1993 12:...	đang học	LH000005
HV000010	Nguyễn Bình Minh	12/03/1992 12:...	đang học	LH000004
NULL	NULL	NULL	NULL	NULL

GIAOVIEN

MaGV	TenGV	NgaySinh	GioiTinh	DienThoai	MaGVQuanLi
GV00001	Nguyễn Văn An	1981-01-02 00:...	Nam	NULL	GV00002
GV00002	Nguyễn Thị Như Lan	1984-12-02 00:...	Nữ	NULL	GV00005
GV00003	Trần Minh Anh	1986-03-23 00:...	Nam	0909123999	GV00002
GV00004	Trương Tường Vi	1988-02-01 00:...	Nữ	0998990909	GV00008
GV00005	Hà Anh Tuấn	1986-12-03 00:...	Nam	0909909000	GV00008
GV00006	Trần Anh Dũng	1979-04-04 00:...	Nam	NULL	GV00010
GV00007	Trần Duy Tân	1978-01-04 00:...	Nam	NULL	GV00002
GV00008	Nguyễn Thị Linh	1979-07-08 00:...	Nữ	0938079700	GV00009
GV00009	Trần Thị Kiều	1977-01-03 00:...	Nữ	NULL	NULL
GV00010	Trần Phương Loan	1978-04-30 00:...	Nữ	NULL	NULL

LOPHOC

MaLop	SiSo	LopTruong	GVQuanLi	NamBatDau	NamKetThuc
LH000001	1	HV000002	GV00001	2010	2014
LH000002	1	HV000003	GV00003	2009	2013
LH000003	2	HV000004	GV00008	2010	2014
LH000004	4	HV000008	GV00010	2011	2015
LH000005	1	HV000009	GV00009	2010	2014

MONHOC

MaMonHoc	TenMonHoc	SoChi
MH00001	Cơ sở dữ liệu	5
MH00002	Cấu trúc dữ liệu	6
MH00003	Mạng máy tính	4
MH00004	Toán cao cấp	6
MH00005	Tin học cơ sở	3
MH00006	Công nghệ phần mềm	4
MH00007	Trí tuệ nhân tạo	4
MH00008	Khai thác dữ liệu	3
MH00009	Phân tích thiết kế hệ thống thông tin	3
MH00010	Hệ thống thông minh	4

KETQUA

MaHV	MaMonHoc	LanThi	Diem
HV000001	MH00001	1	5.5
HV000001	MH00004	1	6
HV000002	MH00001	1	7
HV000002	MH00004	1	8
HV000003	MH00008	1	8.7
HV000003	MH00009	1	9
HV000003	MH00010	1	10
HV000004	MH00008	1	4
HV000004	MH00008	2	3
HV000004	MH00009	1	2
HV000004	MH00009	2	5
HV000004	MH00010	1	6
HV000005	MH00008	1	7.5
HV000005	MH00009	1	1
HV000005	MH00009	2	7
HV000005	MH00010	1	1
HV000005	MH00010	2	3.5

GIAOVIEN_DAY_MONHOC

MaGV	MaMH	ThamNien	SoLopDaDay
GV00001	MH00001	3	*NULL*
GV00001	MH00004	2	*NULL*
GV00002	MH00001	1	*NULL*
GV00002	MH00002	1	*NULL*
GV00003	MH00006	2	*NULL*
GV00003	MH00007	3	*NULL*
GV00003	MH00010	4	*NULL*
GV00004	MH00009	6	*NULL*
GV00004	MH00010	1	*NULL*
GV00005	MH00008	4	*NULL*
GV00005	MH00010	2	*NULL*
GV00006	MH00008	4	*NULL*
GV00006	MH00009	2	*NULL*
GV00006	MH00010	4	*NULL*
GV00007	MH00010	7	*NULL*
GV00008	MH00001	2	*NULL*
GV00008	MH00002	1	*NULL*
GV00009	MH00010	2	*NULL*
GV00010	MH00001	3	*NULL*
GV00010	MH00002	1	*NULL*

PHANCONG

MaGV	MaMH	MaLop
GV00001	MH00001	LH000001
GV00001	MH00004	LH000001
GV00003	MH00010	LH000005
GV00004	MH00009	LH000004
GV00005	MH00008	LH000002
GV00005	MH00008	LH000004
GV00006	MH00008	LH000003
GV00006	MH00009	LH000002
GV00006	MH00009	LH000003
GV00006	MH00010	LH000004
GV00007	MH00010	LH000002
GV00007	MH00010	LH000003
GV00008	MH00002	LH000004

Thực hiện các câu truy vấn sau:

1. Cho biết họ tên và tuổi của các giáo viên nữ.
2. Cho biết họ tên, tình trạng của học viên có mã số 'HV000001'.
3. Cho biết họ tên của các học viên <= 20 tuổi.
4. Cho biết mã số các lớp có sĩ số học viên trong khoảng từ 2 đến 4.
5. Cho biết thông tin giáo viên (mã giáo viên, họ tên, ngày sinh) của các giáo viên mang họ 'Nguyễn'.
6. Cho biết mã số học viên đã từng thi rớt môn 'MH00009' hoặc 'MH00010'.
7. Cho biết họ tên các giáo viên chưa có giáo viên quản lý.
8. Cho biết mã số các giáo viên có quản lý một giáo viên nào đó.
9. Cho biết mã số, họ tên các giáo viên có quản lý một giáo viên nào đó.
10. Cho biết mã số giáo viên quản lý của giáo viên 'GV00006' hoặc 'GV00007' hoặc 'GV00010'.
11. Cho biết mã số các môn học có liên quan đến 'dữ liệu'.
12. Cho biết họ tên các học viên họ 'Trần' đã tốt nghiệp hoặc bị buộc thôi học.
13. Cho biết mã số, họ tên các học viên có làm trưởng lớp một lớp nào đó.
14. Cho biết mã số, tên các môn mà học viên 'HV000005' đã từng thi rớt.

15. Cho biết mã số các học viên từng thi môn ‘MH00009’ nhiều hơn 1 lần.

16. Cho biết Mã họ viên, Tên HV và điểm thi cao nhất môn ‘MH00009’ (liệt kê Mã họ viên, Tên HV và điểm thi cao nhất môn ‘MH00009’).

17. Cho biết họ tên và tuổi của học viên nhỏ tuổi nhất.

18. Cho biết Mã, tên và số tín chỉ của môn học có nhiều học viên từng thi rớt nhất.

19. Cho biết số môn mà học viên ‘Trần Minh An’ đã thi đậu. Học viên được xem là **đậu** một môn khi điểm **lần thi sau cùng** của môn này >= 5.

20. Cho biết điểm trung bình của học viên ‘HV000004’.

Điểm trung bình của học viên chỉ tính trên **lần thi sau cùng** theo công thức:

Điểm trung bình = ∑(Điểm * Số tín chỉ) / ∑Số tín chỉ

21. Cho biết số tín chỉ mà học viên ‘Trần Minh An’ đã đạt được. Số tín chỉ đạt được là tổng số tín chỉ các môn mà học viên đó đã **thi đậu**.

22. Cho biết số tín chỉ mà mỗi học viên đã đạt được. Số tín chỉ đạt được là tổng số tín chỉ các môn mà học viên đó đã thi đậu. Chỉ xuất ra họ tên học viên và số tín chỉ của các học viên đạt nhiều hơn 8 tín chỉ.

23. Xuất ra danh sách học viên cùng điểm trung bình của học viên. Điểm trung bình tính theo công thức câu 20.

24. Cho biết thông tin sinh viên đậu tất cả các môn học trong lần thi đầu tiên.

25. Cho biết thông tin lớp học có tất cả các sinh viên đậu tất cả các môn trong lần thi đầu tiên.

CHƯƠNG 3
LẬP TRÌNH CƠ SỞ DỮ LIỆU BẰNG T-SQL

***Mục tiêu*:**

- *Tìm hiểu về Microsoft Transaction – SQL (T-SQL)*
- *Trình bày được các lệnh T – SQL: biến, If…Else, Case…When, …*
- *Trình bày và sử dụng được các hàm cơ bản trong T – SQL*
- *Mô tả được khái niệm và sử dụng được Trigger để ràng buộc dữ liệu*
- *Mô tả được khái niệm và sử dụng được Stored Procedure*
- *Trình bày được khái niệm và sử dụng được Function và User–Defined Function*
- *Mô tả được khái niệm và sử dụng được kiểu dữ liệu Cursor*

Microsoft Transaction – SQL (T-SQL), là ngôn ngữ cho phép sử dụng các câu lệnh, cú pháp chuẩn SQL để thực hiện các hành động tính toán dữ liệu phức tạp trong Microsoft SQL Server. Các câu lệnh này thông thường sẽ được tổ chức lưu trữ và thực hiện tại máy chủ nhằm cải tiến tốc độ của các ứng dụng theo mô hình khách chủ, tuy nhiên chúng ta cũng có thể thực hiện câu lệnh ở máy Client và gửi về Server để xử lý và trả về kết quả.

3.1. Khai báo và sử dụng biến

Biến trong chương trình là một khái niệm khá quen thuộc khi chúng ta đã làm quen với ngôn ngữ lập trình, biến được dùng để lưu trữ các giá trị tạm thời trong quá trình tính toán các xử lý vì khi thoát khỏi chương trình hay tắt máy thì giá trị của các biến này không còn nữa. Trong T-SQL có 2 loại biến khác nhau: biến cục bộ và biến hệ thống. Do đó trong phần này khi đề cập đến biến thì chúng ta hiểu là biến cục bộ.

3.1.1. Khai báo biến

Khai báo biến là việc chỉ định cho hệ thống máy tính cấp phát một vùng nhớ bên trong bộ nhớ RAM của máy tính để chương trình có thể lưu trữ các giá trị tạm thời trong quá trình tính toán.

Trong T-SQL việc sử dụng biến cần phải được khai báo tường minh rõ ràng, biến phải được khai báo trước rồi mới được phép sử dụng.

Biến có phạm vi hoạt động trong một thủ tục (sẽ tìm hiểu ở phần sau) hoặc trong một bó lệnh (Một bó lệnh là một nhóm các lệnh được được bắt đầu và kết thúc bằng từ khoá GO).

Cú pháp:

```
DECLARE @<tên biến1> <kiểu dữ liệu1>, …
```

Trong đó:

- Tên biến: Là tên của biến được khai báo và luôn bắt đầu bằng ký tự @, tên biến phải duy nhất trong một phạm vi hoạt động.
- Kiểu dữ liệu: Là kiểu dữ liệu cơ bản của Microsoft SQL Server hoặc các kiểu dữ liệu do người dùng định nghĩa, dùng để chỉ định loại dữ liệu mà biến sẽ lưu trữ.

Ví dụ 1: Khai báo 3 biến @masv lưu trữ mã sinh viên, @ngaysinh lưu trữ ngày sinh, @diachi lưu trữ địa chỉ. Trên cửa sổ lệnh ta gõ như sau:

```
Declare @manv char(10), @hoten varchar(30),
        @ngaysinh datetime
```

3.1.2. Gán giá trị cho biến

Sau khi khai báo biến xong, để sử dụng được biến thì phải gán giá trị cho biến. Để gán giá trị vào biến ta sử dụng lệnh SET hoặc lệnh SELECT cùng với phép gán "=".

✍ **Lưu ý**: Ở các thời điểm khác nhau, giá trị của biến có thể khác nhau nhưng tại một thời điểm nhất định thì biến chỉ chứa một giá trị duy nhất.

Cách 1: Dùng lệnh SET:

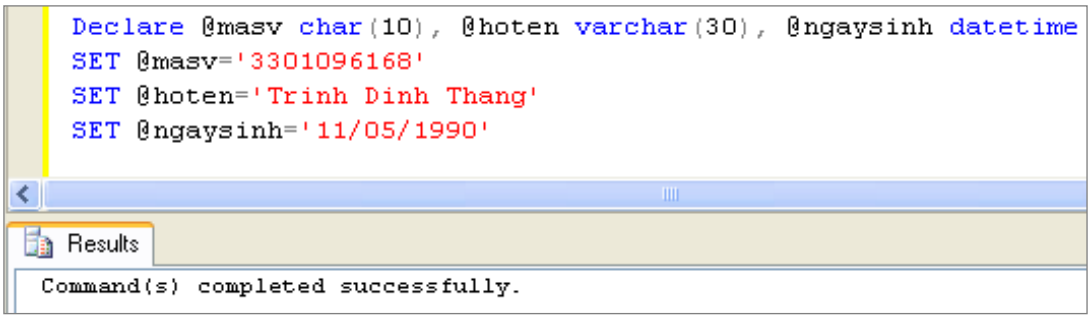
Lệnh SET được dùng để gán giá trị cho một biến. Trường hợp muốn gán nhiều biến thì mỗi lệnh gán phải có lệnh SET tương ứng.

Cú pháp:

SET @<tên biến>=<giá trị>|<biểu thức>|<câu truy vấn>

❖ **Gán giá trị cụ thể cho biến:**

Ví dụ 2: Gán giá trị '3301096168' cho biến @masv, 'Trinh Dinh Thang' cho biến @hoten, '11/05/1990' cho biến @ngaysinh. Thực hiện như sau:



❖ **Gán giá trị của biểu thức cho biến:**

```
Declare @DiemToan Decimal(3,1), @DiemLy Decimal(3,1), @DiemTB Decimal(3,1)
SET @DiemToan=7
SET @DiemLy=8
SET @DiemTB=(@DiemToan+@DiemLy)/2
```

Results

```
Command(s) completed successfully.
```

❖ **Gán câu truy vấn cho biến:**

Ví dụ 3: Gán tổng số sinh viên có địa chỉ ở 'TPHCM' cho biến @TongSV

```
Declare @TongSV int
SET @TongSV=(Select COUNT(MASV)
             From SINHVIEN
             Where DIACHI='TPHCM')
```

Results

```
Command(s) completed successfully.
```

Trong lệnh gán trên, câu truy vấn luôn trả về một giá trị duy nhất (là số sinh viên có địa chỉ ở TPHCM) nên phép gán là hợp lệ. Nếu câu truy vấn được gán cho biến mà trả về nhiều giá trị thì lệnh gán đó không hợp lệ. Ví dụ như trường hợp sau:

Gán Họ tên của sinh viên có địa chỉ ở TPHCM cho biến @Hoten:

```
Declare @Hoten Varchar(30)
SET @Hoten=(Select HOTEN From SINHVIEN
           Where DIACHI='TPHCM')
```

Messages

```
Msg 512, Level 16, State 1, Line 2
Subquery returned more than 1 value. This is not permitted
when the subquery follows =, !=, <, <= , >, >= or when the
subquery is used as an expression.
```

Khi thực thi lệnh gán trên thì ta nhận được lỗi tương ứng "Subquery returned more..." nghĩa là câu truy vấn trả về nhiều giá trị, điều này không được phép gán với các phép toán =,!=,< ...

Cách 2: Dùng lệnh SELECT:

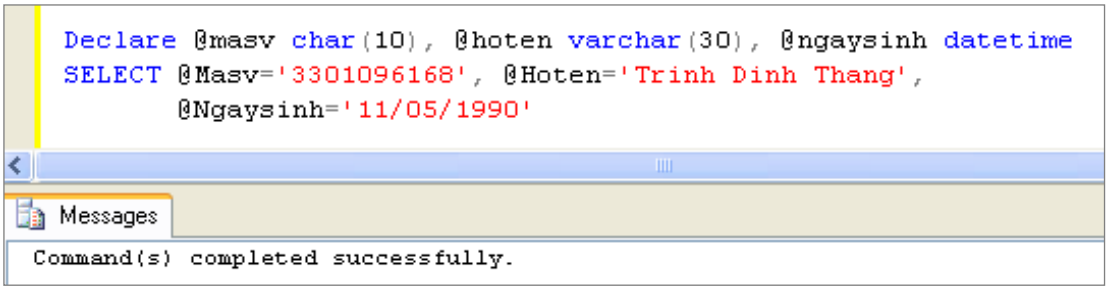
Khắc phục nhược điểm của lệnh SET là mỗi lệnh SET chỉ được sử dụng để gán giá trị cho một biến. Mỗi lệnh SELECT có thể sử dụng để gán giá trị cho nhiều biến cùng lúc.

Cú pháp:

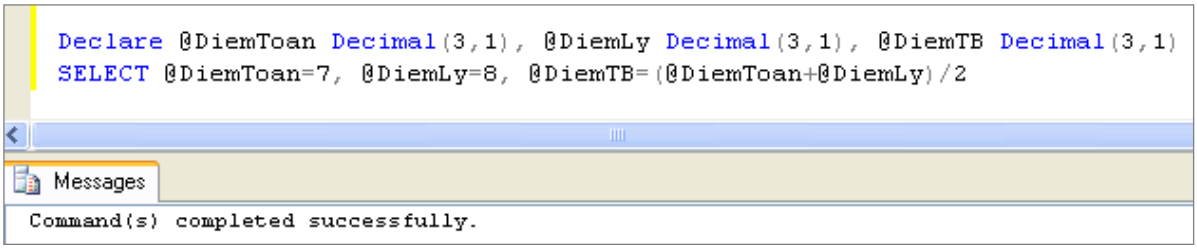
```
SELECT @<tên biến 1>=<giá trị 1>|<biểu thức 1>|<câu truy vấn 1>,
       @<tên biến2>=<giá trị2>|<biểu thức2>|<câu truy vấn2>,
       ...
```

❖ **Gán giá trị cụ thể cho biến:**

Ví dụ 4: Gán giá trị '3301096168' cho biến @Masv, 'Trinh Dinh Thang' cho biến @Hoten, '11/05/1990' cho biến @Ngaysinh. Thực hiện bằng lệnh SELECT như sau:

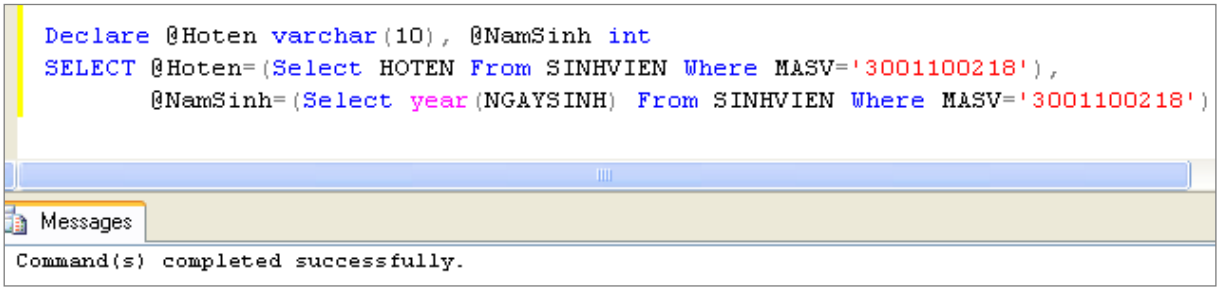


❖ **Gán biểu thức cho biến:**



❖ **Gán câu truy vấn cho biến:**

Ví dụ 5: Gán họ tên và năm của sinh viên có mã là '3001100218' vào 2 biến có tên là @Hoten, @NamSinh



Lệnh gán trên có thể thực hiện ngắn gọn hơn như sau:

```
Declare @Hoten varchar(10), @NamSinh int
SELECT @Hoten=HOTEN, @NamSinh=year(NGAYSINH)
From SINHVIEN
Where MASV='3001100218'
```

Messages

```
Command(s) completed successfully.
```

3.1.3. In giá trị của biến

3.1.3.1. Lệnh PRINT

Lệnh PRINT được dùng để in ra giá trị của một biến duy nhất hoặc một giá trị cụ thể. Trường hợp muốn in ra nhiều giá trị thì phải dùng nhiều lệnh PRINT tương ứng.

Cú pháp:

```
PRINT @<tên biến>|<giá trị cụ thể>|<biểu thức>
```

❖ **In giá trị của các biến:**

Ví dụ 6: In ra giá trị của 3 biến @DiemToan, @DiemLy, @DiemTB

```
Declare @DiemToan Decimal(3,1), @DiemLy Decimal(3,1), @DiemTB Decimal(3,1)
SET @DiemToan=7
SET @DiemLy=8
SET @DiemTB=(@DiemToan+@DiemLy)/2

PRINT @DiemToan
PRINT @DiemLy
PRINT @DiemTB
```

Messages

```
7.0
8.0
7.5
```

❖ **In giá trị cụ thể:**

```
print 23
Print N'Nguyễn Văn Tùng'
Print '02/10/2013'
go
```

Messages

```
23
Nguyễn Văn Tùng
02/10/2013
```

❖ **In giá trị biểu thức:**

```
print 23 + 5
print N'Nguyễn Văn Tùng' + '  ' + N'học lớp 04CDNTH'
```

```
Messages
28
Nguyễn Văn Tùng  học lớp 04CDNTH
```

❖ **In giá trị của biến kết hợp với chuỗi:**

Nếu giá trị biến là kiểu số thì phải chuyển sang kiểu chuỗi bằng cách dùng hàm convert().

```
Declare @DiemToan Decimal(3,1), @DiemLy Decimal(3,1), @DiemTB Decimal(3,1)
SET @DiemToan=7
SET @DiemLy=8
SET @DiemTB=(@DiemToan+@DiemLy)/2

PRINT 'Diem Toan la:'+ convert(char,@DiemToan)
PRINT 'Diem Ly la:'+ convert(char,@DiemLy)
PRINT 'Diem Trung binh la:'+ convert(char,@DiemTB)
```

```
Messages
Diem Toan la:7.0
Diem Ly la:8.0
Diem Trung binh la:7.5
```

3.1.3.2. Lệnh SELECT

Lệnh SELECT cũng được dùng để in ra giá trị của biến. Mỗi lệnh SELECT có thể in ra giá trị cho nhiều biến cùng lúc.

Cú pháp:

```
SELECT @<tên biến1>|<giá trị cụ thể1>|<biểu thức 1>,
       @<tên biến2>|<giá trị cụ thể2>|<biểu thức 2>
            ...
```

Ví dụ 7:

```
Declare @DiemToan Decimal(3,1), @DiemLy Decimal(3,1), @DiemTB Decimal(3,1)
SET @DiemToan=7
SET @DiemLy=8
SET @DiemTB=(@DiemToan+@DiemLy)/2

SELECT @DiemToan as 'Diem Toan', @DiemLy as 'Diem Ly', @DiemTB as 'Diem TB'
```

Results | Messages

	Diem Toan	Diem Ly	Diem TB
1	7.0	8.0	7.5

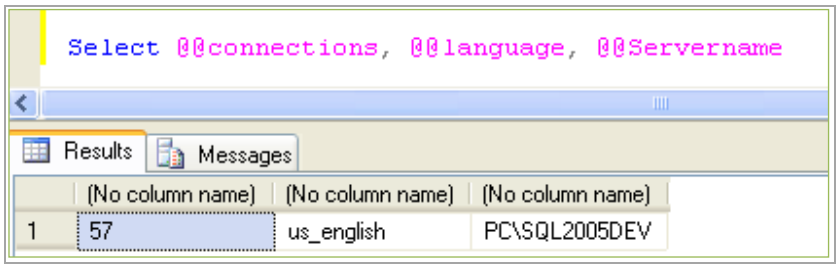
3.1.4. Biến hệ thống

Microsoft SQL Server cung cấp một số biến hệ thống có tên bắt đầu bằng 2 ký tự @, và giá trị mà biến đang lưu trữ là do hệ thống Microsoft SQL Server cung cấp, do đó chúng ta không thể gán giá trị cho các biến này.

Bảng 3.1. Các biến hệ thống thông dụng:

TÊN BIẾN	KIỂU TRẢ VỀ	NỘI DUNG TRẢ VỀ
@@Connections	Số nguyên	Tổng số các kết nối vào SQL Server từ khi nó được khởi động
@@Error	Số nguyên	Số mã lỗi của câu lệnh thực hiện gần nhất. Khi một câu lệnh thực hiện thành công thì biến này có giá trị 0
@@Fetch_status	Số nguyên	Trạng thái của việc đọc dữ liệu trong bảng theo cơ chế từng dòng mẫu tin. Khi đọc dữ liệu của mẫu tin thành công thì biến này có giá trị là 0.
@@Language	Chuỗi	Tên ngôn ngữ mà hệ thống SQL Server đang sử dụng, mặc định là US_English
@@Rowcount	Số nguyên	Tổng số mẫu tin của câu truy vấn gần nhất.
@@Servicename	Chuỗi	Tên dịch vụ hoạt động kèm theo bên dưới SQL Server.
@@Servername	Chuỗi	Tên máy tính cục bộ được cài đặt SQL Server.
@@Version	Chuỗi	Phiên bản, ngày của sản phẩm SQL Server và loại CPU, hệ điều hành cảu máy chủ cài SQL Server.

Để xem giá trị của biến ta dùng lệnh SELECT hoặc lệnh PRINT như sau:



Biến @@Rowcount có giá trị thay đổi tùy theo từng thời điểm chúng ta tác động vào các bảng dữ liệu bằng các câu lệnh truy vấn dữ liệu.

Giả sử khi thực hiện câu lệnh `SELECT * FROM LOP` có kết quả là 3 dòng dữ liệu, sau đó chúng ta xem giá trị của biến @@Rowcount như sau:

```
PRINT @@Rowcount
```

hay

```
SELECT @@Rowcount
```

Kết quả thu được là 3

Với biến @@Language có giá trị luôn cố định là 'US_English'.

3.2. Các toán tử

3.2.1. Toán tử số học

Toán tử số học được sử dụng để tính toán giá trị của biểu thức số, các toán hạng sử dụng trong biểu thức phải có một trong các kiểu dữ liệu sau: int, bigint, smallint, tinyint, numeric, decimal, float, real, money và smallmoney, riêng đối với toán tử chia lấy phần dư chỉ giới hạn sử dụng các kiểu dữ liệu như: int, bigint, smallint và tinyint.

Bảng 3.2. Các toán tử số học :

Toán tử	Ý nghĩa
+	Cộng 2 số
-	Trừ 2 số
*	Nhân 2 số
/	Chia 2 số
%	Chia lấy phần dư

Thứ tự ưu tiên các toán tử là: *, /, %, +, –.

Ví dụ 8:

```
PRINT 2 + 3 * 4 % 5
PRINT 2 + ((3 * 4) % 5)
```

Cả 2 câu lệnh trên đều cho kết quả là 4.

3.2.2. **Toán tử so sánh**

Các toán tử so sánh được dùng để so sánh giá trị của các biểu thức cần so sánh và được sử dụng đối với nhiều kiểu dữ liệu khác nhau như: kiểu số, kiểu chuỗi hoặc kiểu ngày giờ. Thông thường các biểu thức so sánh được sử dụng trong các mệnh đề như: WHERE, HAVING hay các cấu trúc điều khiển như: IF, WHILE…

Bảng 3.3. Các toán tử so sánh:

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!=	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ 9: Sử dụng toán tử so sánh lớn hơn(>) trong mệnh đề WHERE

```
SELECT TENMH
FROM MONHOC
WHERE SOTIET > 30
```

3.2.3. **Các toán tử logic**

Các toán tử logic thường được sử dụng để kết hợp nhiều biểu thức so sánh đơn lẻ thành một biểu thức so sánh chung.

Bảng 3.4. Các toán tử logic:

Toán tử	Ý nghĩa
AND	Và
OR	Hoặc

NOT	Phủ định
BETWEEN	Trong khoảng

Ví dụ 10:

```
SELECT TENSV
FROM SINHVIEN, KETQUA
WHERE SINHVIEN.MASV=KETQUA.MASV AND DIEM > 5
```

3.2.4. Toán tử Bit

Thực hiện thao tác với các bit với các kiểu dữ liệu Integer.

Bảng 3.5. Các toán tử Bit:

Toán tử	Ý nghĩa
&	Thực hiện AND giữa các bit tương ứng của 2 biểu diễn nhị phân.
|	Thực hiện OR giữa các bit tương ứng của 2 biểu diễn nhị phân.
^	Thực hiện XOR giữa các bit tương ứng của 2 biểu diễn nhị phân.
~	Thực hiện NOT của một biểu thức biểu diễn nhị phân.

Ví dụ 11:

```
6 & 69 = 4
```

($6_{10}=100_2$; $69_{10}=1000101_2$; 100_2 AND $1000101_2 = 100_2 = 4_{10}$)

```
6 | 69 = 71
6 ^ 69 = 67
~6 = -7
```

3.3. Các cấu trúc điều khiển

3.3.1. Cấu trúc rẽ nhánh If … Else

Với cấu trúc này, chúng ta có thể chỉ định thực hiện một hoặc nhiều câu lệnh khi giá trị của một biểu thức luận lý là đúng hoặc sai. Cấu trúc rẽ nhánh được phép sử dụng bên trong một bó lệnh (batch) hoặc bên trong một thủ tục, hàm hay trigger. Cấu

trúc rẽ nhánh được phép lồng nhiều cấp bên trong và cấp độ lồng nhau này là không giới hạn.

Cú pháp:

```
IF < điều kiện 1 >
    < khối lệnh 1 >
ELSE IF < điều kiện 2 >
    < khối lệnh 2 >
    …
ELSE
    < khối lệnh n >
```

Trong đó:

Điều kiện: Thông thường là một biểu thức luận lý để chỉ một điều kiện so sánh nào đó.

Khối lệnh: Các lệnh sẽ được thực hiện khi biểu thức so sánh có giá trị đúng (True).

Ví dụ 12: Khai báo và gán giá trị cho biến tuổi, sau đó kiểm tra:

Nếu Tuổi <=18: in ra ‘Còn nhỏ’

Nếu 18< Tuổi <=35: in ra ‘Thanh niên’

Nếu 35< Tuổi <=55: in ra ‘Trung niên’

Nếu Tuổi > 55: in ra ‘Già’

```
Declare @tuoi int
set @tuoi=60
if(@tuoi<=18)
    print N'Còn nhỏ'
else if(@tuoi<=35)
    print N'Thanh niên'
else if(@tuoi<=55)
    print N'Trung niên'
else
    print N'Già'
```

Ví dụ 13: Cho MONHOC(MAMH, TENMH, SOTC, LOAIMH)

Viết lệnh kiểm tra nếu LOAIMH của môn học có MAMH ‘01300014’ là ‘BAT BUOC’ thì in ra chuỗi ‘Mon hoc bat buoc’, ngược lại in ra chuỗi ‘Mon hoc tu chon’

```
IF((Select LOAIMH FROM MONHOC WHERE MAMH='01300014')='BAT BUOC')
    PRINT 'Mon hoc bat buoc'
ELSE
    PRINT 'Mon hoc tu chon'
```

Ngoài ra chúng ta có thể sử dụng từ khoá EXISTS kết hợp với cấu trúc IF để kiểm tra sự tồn tại của các dòng dữ liệu trong bảng một cách hiệu quả hơn.

Cú pháp:

```
IF EXISTS (Câu lệnh SELECT)
    <Câu lệnh 1>|<Khối lệnh 1>
[ELSE
    <Câu lệnh 2>|<Khối lệnh 2>]
```

Trong đó:

Từ khoá EXISTS: Dùng để kiểm tra sự tồn tại các dòng dữ liệu trong câu lệnh SELECT sau nó. Kết quả IF trả về đúng (TRUE) khi câu lệnh SELECT trả về ít nhất một dòng dữ liệu, ngược lại trả về sai (FASLE).

Ví dụ 14: SINHVIEN(MASV, HOTEN, GIOITINH, DIACHI, MALOP, DTB)

Kiểm tra xem có sinh viên nào có địa chỉ ở KHÁNH HÒA không, nếu có thì in ra danh sách sinh viên đó, nếu không thì in ra thông báo 'Không có sinh viên nào ở KHÁNH HÒA'

```
if exists(select * from sinhvien where quequan like N'%KHÁNH HÒA')
  select * from sinhvien where quequan like N'%KHÁNH HÒA'

else
  print N'Không có sinh viên nào ở KHÁNH HÒA'
```

3.3.2. Cấu trúc Case

Cấu trúc này được dùng để đánh giá một biểu thức và trả về một hoặc một số các kết quả dựa vào giá trị của biểu thức. Có 2 cú pháp Case khác nhau như sau:

Cú pháp 1: một biểu thức sẽ được dùng để so sánh với một tập các giá trị để xác định kết quả.

```
CASE <Biểu thức>
    WHEN <Giá trị> THEN <Kết quả>
        […n]
    [ELSE <Kết quả khác>
END
```

Ví dụ 15: Cho NHANVIEN(MANV, HOTEN, NGAYSINH, GIOITINH, DIACHI)

Cho biết những nhân viên đến tuổi nghỉ hưu biết rằng tuổi nghỉ hưu của nam là 60, tuổi nghỉ hưu của nữ là 55

```
SELECT *
FROM NHANVIEN
WHERE DATEDIFF(YY,NGAYSINH,GETDATE()) >= CASE GIOITINH
                                          WHEN 'NAM' THEN 60
                                          WHEN 'NU' THEN 55
                                          END
```

Cú pháp 2: đánh giá tập các biểu thức điều kiện để xác định kết quả

```
CASE
    WHEN <Biểu thức điều kiện> THEN <Kết quả 1>
        [… n]
    [ELSE <Kết quả khác>
END
```

Ví dụ 16: SINHVIEN(MASV, HOTEN, GIOITINH, DIACHI, MALOP, DTB)

In ra Mã sinh viên, họ tên sinh viên, điểm trung bình, xếp loại của các sinh viên. Trong đó xếp loại được tính dựa trên điểm trung bình như sau:

DTB >= 8 : Giỏi

7<= DTB < 8 : Khá

5<= DTB < 7: Trung bình

DTB <5 : Yếu

```
SELECT masv, hoten, dtb, N'Xếp loại' = CASE
                        WHEN dtb>=8 THEN N'Giỏi'
```

```
                    WHEN dtb>=7 THEN N'Khá'
                    WHEN dtb>=5 THEN N'Trung bình'
                    ELSE N'Yếu'
                    END
FROM SINHVIEN
```

3.3.3. Cấu trúc lặp While

Với cấu trúc lặp WHILE, chúng ta có thể chỉ định một hoặc nhiều câu lệnh sẽ được thực hiện lặp lại nhiều lần trong khi giá trị của biểu thức điều kiện vẫn còn đúng. Giống như cấu trúc rẽ nhánh, cấu trúc lặp được phép sử dụng bên trong một bó lệnh (Batch) hoặc bên trong các thủ tục, hàm, trigger. Giữa cấu trúc rẽ nhánh và cấu trúc lặp không có thứ tự ưu tiên khi chúng lồng vào nhau và cấp độ lồng nhau là không có giới hạn.

Cú pháp:

```
WHILE <Biểu thức điều kiện>
BEGIN
    <Các lệnh >
END
```

Có thể sử dụng lệnh BREAK và CONTINUE trong khối lệnh WHILE:

- Break: thoát khỏi vòng While.
- Continue: bỏ qua các lệnh sau nó, trở lại đầu vòng While

Ví dụ 17: In ra các số nguyên từ 1 đến 10

```
DECLARE @a int
SET @a=1
WHILE(@a<=10)
BEGIN
        PRINT @a
        SET @a=@a+1
END
```

Ví dụ 18: Ví dụ sau in ra tổng 2 số nguyên a và b cho tới khi nào tổng này bằng 9 thì dừng lại và thoát khỏi vòng lặp.

```
DECLARE @a int
DECLARE @b int
SET @a=1
SET @b=2
WHILE(@a<10)and (@b<10)
BEGIN
        PRINT @a+@b
        IF(@a+@b=9)
                BREAK
        SET @a=@a+1
        SET @b=@b+2
END
```

Thông thường cấu trúc WHILE được ứng dụng hiệu quả để duyệt qua từng mẫu tin trong bảng thông qua cấu trúc Cursor (sẽ được trình bày trong phần sau).

3.3.4. Cấu trúc WaitFor

Cấu trúc này được dùng để ngăn chặn một query batch, một thủ tục, một giao dịch đến thời điểm nào đó hoặc sau một khoảng thời gian nào đó.

Cú pháp:

```
WAITFOR  DELAY 'time_to_pass'|TIME 'time_to_execute'
```

Ví dụ 19: Chờ đến 7:30PM thì in ra câu thông báo ‘Đến giờ rồi’

```
WAITFOR TIME '7:30PM'
PRINT N'Đến giờ rồi'
```

3.4. Các hàm thông dụng

3.4.1. Các hàm xử lý chuỗi

Các hàm xử lý chuỗi thường có tham số đầu vào là kiểu dữ liệu chuỗi và giá trị trả về của chúng cũng là kiểu dữ liệu chuỗi hoặc kiểu dữ liệu số.

3.4.1.1. Hàm UPPER

Hàm UPPER được đùng để chuyển đổi một chuỗi ký tự thành chữ in

Cú pháp: `UPPER(<chuỗi dữ liệu>)`

Ví dụ 20:

```
select upper(N'Nguyễn Thanh Hoàng') as 'IN HOA'
```

Results | Messages

	IN HOA
1	NGUYỄN THANH HOÀNG

Ví dụ 21: Cho SINHVIEN(MASV, HOTEN, GIOITINH, DIACHI, MALOP, DTB). In danh sách họ tên sinh viên bằng chữ in

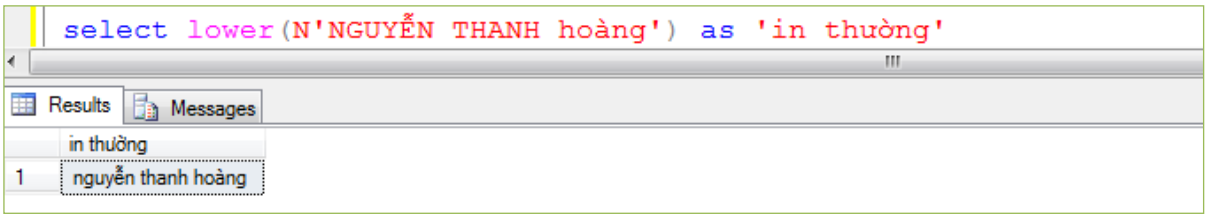
```
SELECT N'Họ tên SV'=UPPER(HOTEN) FROM SINHVIEN
```

3.4.1.2. Hàm LOWER

Hàm LOWER được đùng để chuyển đổi một chuỗi ký tự thành chữ thường.

Cú pháp: `LOWER(<chuỗi dữ liệu>)`

Ví dụ 22:



Ví dụ 23: Cho SINHVIEN(MASV, HOTEN, GIOITINH, DIACHI, MALOP, DTB)

In danh sách họ tên sinh viên bằng chữ thường.

```
SELECT N'Họ tên SV'=LOWER(HOTEN) FROM SINHVIEN
```

3.4.1.3. Các hàm LEFT, RIGHT, SUBSTRING

Các hàm này có kết quả trả về là một chuỗi con được trích ra từ một chuỗi nguồn cho trước. Chuỗi con được trích ra bắt đầu từ vị trí bên trái (LEFT), bên phải (RIGHT), hoặc tại bất kỳ một vị trí nào (SUBSTRING).

Cú pháp:

```
LEFT(<chuỗi nguồn>,<số ký tự>)
RIGHT(<chuỗi nguồn>,<số ký tự>)
SUBSTRING(<chuỗi nguồn>,<vị trí>,<số ký tự>)
```

Trong đó:

- Chuỗi nguồn: Là chuỗi ký tự nguồn.
- Số ký tự: Là một số nguyên dương chỉ định số ký tự bên trong chuỗi nguồn sẽ được trích ra.
- Vị trí: Là số nguyên dương chỉ định vị trí bắt đầu trích chuỗi (áp dụng cho hàm SUBSTRING).

Ví dụ 24: Sử dụng các hàm trích chuỗi nguồn "Microsoft SQL Server"

```
PRINT LEFT('MICROSOFT SQL SERVER',9)
PRINT RIGHT('MICROSOFT SQL SERVER',6)
PRINT SUBSTRING('MICROSOFT SQL SERVER',11,3)
```

```
Messages
MICROSOFT
SERVER
SQL
```

3.4.1.4. Các hàm LTRIM, RTRIM

Các hàm này có kết quả trả về là một chuỗi đã cắt bỏ khoảng trắng ở đầu chuỗi (LTRIM) hoặc các khoảng trắng ở cuối chuỗi (RTRIM).

Cú pháp:

```
LTRIM(<chuỗi dữ liệu>)
RTRIM(<chuỗi dữ liệu>)
```

Ví dụ 25:

```
PRINT LTRIM('     MICROSOFT SQL SERVER')
PRINT RTRIM('MICROSOFT SQL SERVER      ')
```

```
Messages
MICROSOFT SQL SERVER
MICROSOFT SQL SERVER
```

3.4.1.5. Hàm SPACE

Hàm SPACE có kết quả trả về là một chuỗi chứa N ký tự trắng.

Cú pháp: `SPACE(N)`

Trong đó: N: Là một số nguyên dương dùng để chỉ định chuỗi chứa bao nhiêu ký tự trắng.

Ví dụ: Sử dụng hàm SPACE để tạo ký tự trắng như sau:

```
PRINT 'MICROSOFT' + SPACE(2) + 'SQL' + SPACE(2) + 'SERVER'
```

```
Messages
MICROSOFT  SQL  SERVER
```

3.4.1.6. Hàm REPLICATE

Hàm này có kết quả trả về là một chuỗi chứa các ký tự được lặp lại N lần.

Cú pháp: `REPLICATE(<chuỗi lặp>, N)`

Trong đó:

– Chuỗi lặp: Là một chuỗi có các ký tự sẽ được lặp lại.

– N: Là một số nguyên dương chỉ định số lần lặp lại.

Ví dụ 26: Thực hiện lặp lại chuỗi ‘Hello’ 3 lần:

```
PRINT REPLICATE('HELLO!   ',3)
```

```
Messages
HELLO!   HELLO!   HELLO!
```

3.4.1.7. Hàm LEN

Hàm LEN được sử dụng để lấy chiều dài của một chuỗi. Kết quả trả về là một số nguyên dương cho biết chiều dài chuỗi.

Cú pháp: `LEN(<chuỗi dữ liệu>)`

Ví dụ 27: Tính chiều dài của chuỗi ‘MICROSOFT SQL SERVER’

```
PRINT LEN('MICROSOFT SQL SERVER')
```

```
Messages
20
```

3.4.1.8. Hàm REVERSE

Hàm REVERSE được dùng để đảo ngược một chuỗi cho trước.

Cú pháp: `REVERSE(<chuỗi dữ liệu>)`

Ví dụ 28: Đảo ngược chuỗi sau: ‘MICROSOFT SQL SERVER’

```
PRINT REVERSE('MICROSOFT SQL SERVER')
```

```
Messages
REVRES LQS TFOSORCIM
```

3.4.1.9. Hàm STUFF

Hàm STUFF có kết quả trả về là một chuỗi mới sau khi đã huỷ bỏ một số ký tự hiện có và thêm vào một chuỗi con khác tại vị trí vừa huỷ bỏ.

Cú pháp:

`STUFF(<chuỗi nguồn>,<vị trí>,<chiều dài>,<chuỗi con>)`

Trong đó:

– Chuỗi nguồn: Là một chuỗi dữ liệu.

– Vị trí: Là một số nguyên chỉ định vị trí bắt đầu huỷ bỏ các ký tự bên trong chuỗi nguồn.

– Chiều dài: Là một số nguyên chỉ định bao nhiêu ký tự sẽ bị huỷ bỏ bên trong chuỗi nguồn được đếm từ ký tự đầu tiên bên trái chuỗi nguồn.

– Chuỗi con: Là một chuỗi sẽ được thêm vào chuỗi nguồn tại vị trí huỷ bỏ ở trên.

Ví dụ 29: hàm STUFF sau hủy bỏ 11 ký tự từ vị trí thứ 10 của chuỗi 'MICROSOFT SQL SERVER' và thêm vào chuỗi ' OFFICE' vào vị trí vừa hủy bỏ:

```
PRINT STUFF('MICROSOFT SQL SERVER',10,11,' OFFICE')
```

```
Messages
MICROSOFT OFFICE
```

3.4.1.10. Hàm REPLACE

Hàm REPLACE cho phép tìm và thay thế (nếu có) một chuỗi con trong chuỗi nguồn bằng một chuỗi khác.

Cú pháp:

```
REPLACE(<chuỗi nguồn>, <chuỗi tìm>, <chuỗi thay thế>)
```

Trong đó:

– Chuỗi tìm: Là chuỗi con cần tìm xem có xuất hiện trong chuỗi nguồn hay không.

– Chuỗi thay thế: Là chuỗi sẽ được thay thế khi tìm thấy chuỗi tìm trong chuỗi nguồn.

Ví dụ 30: Thực hiện việc tìm kiếm và thay thế như sau:

```
PRINT REPLACE('MICROSOFT SQL SERVER','SQL SERVER','OFFICE')
```

```
Messages
MICROSOFT OFFICE
```

3.4.1.11. Hàm CHAR

Hàm CHAR có kết quả trả về là một ký tự tương ứng trong bảng mã ASCII

Cú pháp: `CHAR(số nguyên)`

Ví dụ 31:

```
PRINT CHAR(65)
```

```
Messages
A
```

3.4.1.12. Hàm ASCII

Hàm ASCII có kết quả trả về là một số nguyên có phạm vi từ 0-255 tương ứng trong bảng mã ASCII, tham số đầu vào là một ký tự.

Cú pháp: `ASCII(<ký tự>)`

Ví dụ 32:

```
PRINT ASCII('A')
```

```
Messages
65
```

3.4.2. Các hàm toán học

Các hàm toán học thường được dùng để xử lý các dữ liệu kiểu số, tham số đầu vào thường là kiểu dữ liệu số và giá trị trả về của chúng cũng là kiểu dữ liệu số.

3.4.2.1. Hàm ABS

Hàm ABS được dùng để lấy trị tuyệt đối của một số, kết quả trả về là một số dương.

Cú pháp: `ABS(<biểu thức số>)`

Trong đó:

- Biểu thức số: Là một biểu thức có kiểu dữ liệu số hoặc một giá trị số cụ thể mà chúng ta muốn tính giá trị tuyệt đối.

Ví dụ 33: Thực hiện lấy trị tuyệt đối của các biểu thức sau:

```
PRINT ABS(-5)
PRINT ABS(20 +(6-30))
```

```
Messages
5
4
```

3.4.2.2. Hàm PI

Hàm PI có kết quả trả về là hằng số PI trong toán học.

Cú pháp: `PI()`

Ví dụ 34:

```
PRINT PI()
```

```
Messages
3.14159
```

3.4.2.3. Hàm POWER

Hàm POWER được dùng để tính luỹ thừa của một số nào đó theo một số mũ chỉ định trước.

Cú pháp: `POWER(<biểu thức số>, <số mũ>)`

Trong đó:

- Biểu thức số: Là một biểu thức có giá trị là kiểu dữ liệu số.
- Số mũ: Là một số dương.

Ví dụ 35:

```
PRINT POWER(3,4)
```

```
Messages
81
```

3.4.2.4. Hàm RAND

Hàm RAND có kết quả trả về là một số thực ngẫu nhiên mà hệ thống Microsoft SQL Server tự động tạo ra đảm bảo không trùng lắp.

Cú pháp: `RAND([số nguồn])`

Trong đó:

- Số nguồn: Là một số ngẫu nhiên có giá trị không vượt quá phạm vi của kiểu dữ liệu int làm giá trị nguồn cho hệ thống tạo ra số ngẫu nhiên.

Ví dụ 36:

```
PRINT RAND(12)
```

```
Messages
0.713797
```

3.4.2.5. Hàm ROUND

Hàm ROUND được sử dụng để làm tròn một số.

Cú pháp: `ROUND(<biểu thức số>, <Vị trí làm tròn>)`

Trong đó:

- Biểu thức số: Là một biểu thức có kiểu dữ liệu là số thực.
- Vị trí làm tròn: Là một số nguyên âm hoặc dương dùng để chỉ định vị trí muốn làm tròn, được tính từ vị trí dấu chấm thập phân.

Ví dụ 37: Thực hiện làm tròn số 243.6531 với các hình thức sau:

```
PRINT ROUND(243.6531,1)
PRINT ROUND(243.6531,2)
PRINT ROUND(243.6531,-1)
```

```
Messages
243.7000
243.6500
240.0000
```

3.4.2.6. Hàm SIGN

Hàm SIGN có kết quả trả về là một số qui định dấu của một biểu thức số. Nếu giá trị trả về là 1 thì biểu thức số có giá trị dương, nếu là -1 thì biểu thức số âm, nếu là 0 thì biểu thức số bằng 0.

Cú pháp: `SIGN(<biểu thức số>)`

Ví dụ 38:

```
PRINT SIGN((20-10)*2 + 3)
PRINT SIGN((10-20)*2 + 3)
PRINT SIGN((10-20)*2 + 20)
```

```
Messages
1
-1
0
```

3.4.2.7. Hàm SQRT

Hàm SQRT được dùng để tính căn bậc 2 của một số bất kỳ và kết quả trả về là một số dương.

Cú pháp: `SQRT(<biểu thức số>)`

Ví dụ 39:

```
PRINT SQRT(9)
```

```
Messages
3
```

3.4.2.8. Hàm FLOOR

Hàm Floor trả về số nguyên lớn nhất nhỏ hơn hoặc bằng số nguồn trong cú pháp sau:

Cú pháp: `FLOOR(<số nguồn>)`

Ví dụ 40:

```
PRINT FLOOR(3.3)
```

Messages

3

3.4.2.9. Hàm CEILING

Hàm Ceiling trả về số nguyên nhỏ nhất lớn hơn hoặc bằng số nguồn trong cú pháp sau:

Cú pháp: `CEILING(<số nguồn>)`

Ví dụ 41:

```
PRINT CEILING(3.3)
```

Messages

4

3.4.3. Các hàm xử lý ngày giờ

Để xử lý dữ liệu liên quan đến ngày, giờ, hệ thống Microsoft SQL Server cung cấp một số hàm thông dụng đáp ứng được hầu hết các yêu cầu của người sử dụng cơ sở dữ liệu. Các hàm này thường có tham số vào là kiểu dữ liệu ngày giờ và giá trị trả về của chúng có thể là kiểu dữ liệu số, chuỗi hoặc ngày giờ.

Bảng 3.6. Mô tả viết tắt của các đơn vị thời gian:

Từ viết tắt	Ý nghĩa	Miền giá trị
yy	Năm	1900 – 9999
qq	Quí	1 – 4
mm	Tháng	1 – 12
dd	Ngày trong tháng	1 – 31
dy	Ngày trong năm	1 – 366
wk	Tuần	1 – 53
dw	Ngày trong tuần	1 – 7
hh	Giờ trong ngày	0 – 23

mi	Phút trong giờ	0 – 59
ss	Giây trong phút	0 – 59
ms	Phần trăm mili giây	0 - 999

3.4.3.1 Hàm DATEADD

Hàm DATEADD cho phép cộng hoặc trừ một ngày chỉ định cho một đơn vị thời gian nào đó và trả về một ngày mới theo yêu cầu.

Cú pháp:

```
DATEADD(<đơn vị thời gian>,<số nguyên>,<ngàychỉđịnh>)
```

Trong đó:

- Đơn vị thời gian: Là đơn vị dùng cho việc giảm hoặc tăng ngày (dd), tháng (mm), năm (yy)…
- Số nguyên: Là một số nguyên âm hoặc dương chỉ định việc giảm hoặc tăng theo đơn vị thời gian.
- Ngày chỉ định: Là một biểu thức, tên cột dữ liệu, giá trị cụ thể có kiểu dữ liệu ngày.

Ví dụ 42:

❖ Tăng thêm 10 ngày đối với ngày 20/03/2014

```
PRINT DATEADD(dd, 10, '03/20/2014')
```

```
Messages
Mar 30 2014 12:00AM
```

❖ Tăng thêm 10 tháng đối với ngày 20/03/2014

```
PRINT DATEADD(mm, 10, '03/20/2014')
```

```
Messages
Jan 20 2015 12:00AM
```

❖ Tăng thêm 10 năm đối với ngày 20/03/2014

```
PRINT DATEADD(yy, 10, '03/20/2014')
```

```
Messages
Mar 20 2024 12:00AM
```

3.4.3.2 Hàm DATEDIFF

Hàm DATEDIFF được dùng để tính khoảng cách (hiệu) của 2 ngày bất kỳ và có kết quả trả về là một số nguyên.

Cú pháp:

```
DATEDIFF(<đơn vị thời gian>, <ngày 1>, <ngày 2>)
```

Trong đó:

- Đơn vị thời gian: Là đơn vị có thể là ngày (dd), tháng (mm), năm (yy)…
- Ngày 1, ngày 2: Là các biểu thức, tên cột dữ liệu, giá trị cụ thể có kiểu dữ liệu ngày.

Ví dụ 43: Tính hiệu giữa 2 ngày: 20/05/2014 và 15/06/2014

```
PRINT DATEDIFF(dd, '05/20/2014' , '07/15/2014')
PRINT DATEDIFF(mm, '05/20/2014' , '07/15/2014')
PRINT DATEDIFF(yy, '05/20/2014' , '07/15/2014')
```

Messages

```
56
2
0
```

3.4.3.3 Hàm GETDATE

Hàm GETDATE được dùng để lấy ngày giờ hiện tại theo ngày giờ của hệ thống .

Cú pháp : `GETDATE()`

Ví dụ 44: Để lấy ngày hiện hành chúng ta sử dụng hàm GETDATE như sau:

```
PRINT GETDATE()
```

Messages

```
Jan  2 2014 11:50AM
```

3.4.3.4 Hàm DATENAME

Hàm DATENAME có kết quả trả về là một chuỗi thời gian đại diện của một ngày chỉ định theo một đơn vị thời gian bất kỳ.

Cú pháp: `DATENAME(<đơn vị thời gian>,<ngày chỉ định>)`

Trong đó:

- Đơn vị thời gian: Là đơn vị dùng để chỉ định lấy ra chuỗi thời gian đại diện.
- Ngày chỉ định: Là một biểu thức, tên cột dữ liệu hoặc giá trị cụ thể có kiểu dữ liệu ngày.

Ví dụ 45:

```
PRINT DATENAME(dw,'03/20/2014')
PRINT DATENAME(dd,'03/20/2014')
PRINT DATENAME(mm,'03/20/2014')
PRINT DATENAME(yy,'03/20/2014')
```

```
Messages
Thursday
20
March
2014
```

3.4.3.5 Hàm DATEPART

Hàm DATEPART trả về giá trị của thành phần đơn vị thời gian trong ngày.

Cú pháp: `DATEPART(<đơn vị thời gian>,<ngày chỉ định>)`

Ví dụ 46:

```
PRINT DATEPART(dw,'03/20/2014')
PRINT DATEPART(dd,'03/20/2014')
PRINT DATEPART(mm,'03/20/2014')
PRINT DATEPART(yy,'03/20/2014')
```

```
Messages
5
20
3
2014
```

3.4.3.6 Các hàm DAY, MONTH, YEAR

Các hàm DAY, MONTH, YEAR có kết quả trả về là một số nguyên chỉ định ngày (day), tháng (month), năm (year) của một ngày bất kỳ.

Cú pháp:

```
DAY(<ngày chỉ định>)
MONTH(<ngày chỉ định>)
YEAR(<ngày chỉ định>)
```

Ví dụ 47:

```
PRINT DAY('03/20/2014')
PRINT MONTH('03/20/2014')
PRINT YEAR('03/20/2014')
```

```
Messages
20
3
2014
```

3.4.4. Các hàm chuyển đổi kiểu dữ liệu

Các hàm chuyển đổi kiểu dữ liệu được dùng để chuyển đổi qua lại các kiểu dữ liệu tương thích nhau bên trong Microsoft SQL Server. Thông thường trong các xử lý, chúng ta thường chuyển đổi các kiểu dữ liệu số hoặc kiểu dữ liệu ngày giờ về kiểu dữ liệu chuỗi để hiển thị ra màn hình. Đối với các kiểu dữ liệu image, text, ntext rất hạn chế trong việc chuyển đổi qua lại các kiểu dữ liệu khác.

3.4.4.1 Hàm CAST

Hàm CAST cho phép chúng ta chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kỳ theo yêu cầu.

Cú pháp: `CAST (<biểu thức> AS <kiểu dữ liệu>)`

Trong đó:

- Biểu thức: Là tên của một cột trong bảng hoặc một biểu thức tính toán cần chuyển sang kiểu dữ liệu mới.
- Kiểu dữ liệu: Là tên kiểu dữ liệu mới mà biểu thức sẽ được chuyển đổi sang.

Ví dụ 48: Chuyển đổi số 123 sang kiểu dữ liệu Varchar

```
PRINT CAST(123 AS VARCHAR(3))
```

3.4.4.2 Hàm CONVERT

Hàm CONVERT cho phép chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kỳ như mong muốn và có thể theo một định dạng nào đó.

Cú pháp:

```
CONVERT(<kiểudữliệu>[(length)],<biểuthức>[,địnhdạng])
```

Trong đó:

- Kiểu dữ liệu: Là tên kiểu mới cần được chuyển đổi sang.
- Biểu thức: Là tên của cột bên trong bảng hoặc một biểu thức tính toán muốn chuyển sang kiểu dữ liệu mới.
- Định dạng: Là một số nguyên dương chỉ định việc chuyển đổi kiểu dữ liệu ngày sang dạng chuỗi.

Bảng 3.7. Mô tả một số định dạng thường dùng trong hàm CONVERT:

Định dạng năm (yy)	Định dạng năm (yyyy)	Dạng hiển thị dữ liệu
1	101	mm/dd/yy
2	102	yy.mm.dd

3	103	dd/mm/yy
4	104	dd.mm.yy
5	105	dd-mm-yy
6	106	dd mon yy
7	107	mon dd yy
8	108	hh:mm:ss
9	109	mon dd yyyy hh:mm:ss
10	110	mm-dd-yy
11	111	yy/mm/dd
12	112	Yymmdd
13	113	dd mon yyyy hh:mm:ss
14	114	hh:mm:ss:mmm
	21 hoặc 121	yyyy-mm-dd hh:mi:ss.mmm
	20 hoặc 120	yyyy-mm-dd hh:mi:ss

Ví dụ 49: In ra họ tên và ngày sinh của sinh viên, ngày sinh được định dạng "dd/mm/yyyy"

```
SELECT HOTEN , CONVERT(VARCHAR(10), NGSINH,103)
FROM SINHVIEN
```

3.4.4.3 Hàm STR

Hàm STR được dùng để chuyển đổi kiểu dữ liệu số sang kiểu dữ liệu chuỗi. Phải đảm bảo vùng trắng để chứa các ký số khi chuyển đổi sang kiểu dữ liệu chuỗi.

Cú pháp: `STR(<số thực>,<số ký tự>[,số lẻ])`

Trong đó:

- Số thực: Là một biểu thức có kiểu dữ liệu số thực.
- Số ký tự: Số khoảng trắng dùng để chứa các ký số sau khi chuyển sang kiểu dữ liệu chuỗi.

– Số lẻ: Chỉ định số thập phân

Ví dụ 50: `PRINT STR(12.5, 10, 2)`

Kết quả sẽ là: $\underbrace{\_\_\_\_\_12.\overbrace{50}^{2}}_{10}$

Ví dụ 51: In danh sách sinh viên (MASV, HOTEN), mã môn học (MAMH) và kết quả học tập (DIEM) của từng sinh viên trong đó cột điểm được định dạng có 1 chữ số thập phân.

```
SELECT SINHVIEN.MASV,HOTEN,MAMH,DIEM_MH=STR(DIEM,4,1)
FROM SINHVIEN, KETQUA
WHERE KETQUA.MASV=SINHVIEN.MASV
```

Kết quả trả về như sau:

	MASV	HOTEN	MAMH	DIEM_MH
1	SV1	Nguyen Tuan Anh	m01	6.0
2	SV2	Tran Thanh Nam	m01	8.0
3	SV2	Tran Thanh Nam	m02	7.0
4	SV3	Duong Minh Tuan	m01	9.0
5	SV3	Duong Minh Tuan	m02	4.0
6	SV3	Duong Minh Tuan	m03	5.0

3.5. Thủ tục thường trú (Stored Procedure - SP)

Thủ tục thường trú là một tập hợp các dòng lệnh, các biến và các cấu trúc điều khiển trong ngôn ngữ T-SQL dùng để thực hiện một hành động nào đó. Nội dung của SP sẽ được lưu trữ tại cơ sở dữ liệu của Microsoft SQL Server.

SP cũng có những tính chất giống như thủ tục trong một số ngôn ngữ lập trình như: mỗi thủ tục đều có tên, có thể có tham số truyền vào, tham số trả giá trị ra. Ngoài ra bên trong thân của một SP chúng ta cũng có thể gọi thực thi một SP khác đã được tạo trước đó. Phạm vi hoạt động của các SP là cục bộ bên trong một CSDL lưu trữ thủ tục đó.

SP còn có thể được gọi thực hiện trong môi trường không phải là Microsoft SQL Server, đây là nét đặc biệt của SP. Do đó khi xây dựng các màn hình giao diện trên các ngôn ngữ lập trình khác thì chúng ta vẫn có thể gọi SP thực hiện.

Lợi ích của thủ tục:

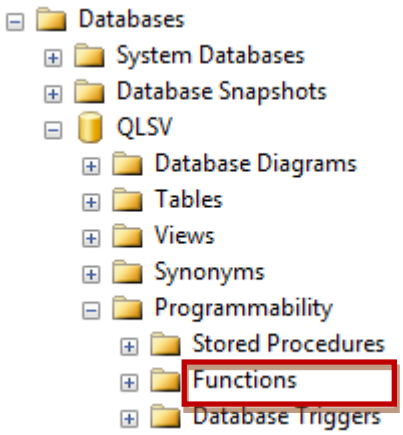
– Chúng ta chỉ tạo thủ tục một lần và lưu trữ trong CSDL, sau đó có thể gọi nó với số lần bất kỳ.

– Thủ tục cho phép thực thi nhanh hơn: nếu một xử lý yêu cầu một đoạn T-SQL khá lớn thì thủ tục thực thi nhanh hơn một loạt các lệnh T-SQL. Chúng được phân tích cú pháp và tối ưu hóa trong lần thực thi đầu tiên và một phiên bản dịch của chúng sẽ được lưu trong bộ nhớ để sử dụng cho các lần sau, nghĩa là những lần sau chúng chỉ sử dụng kết quả của lần biên dịch đầu tiên. Vì vậy trong các xử lý chúng ta nên dùng thủ tục để tăng tốc độ xử lý.

– Thủ tục có thể giảm bớt vấn đề kẹt mạng: giả sử một xử lý có hàng trăm lệnh T-SQL và được thực hiện thông qua từng dòng lệnh đơn lẻ. Khi thực hiện, ta phải gửi hàng trăm lệnh đó lên mạng và gây kẹt mạng. Rõ ràng, phương án tạo ra một thủ tục là tối ưu hơn.

– Thủ tục hữu ích trong vấn đề bảo mật: thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, bằng cách phân quyền cho một người dùng được phép sử dụng các thủ tục nào và không được phép sử dụng các thủ tục nào.

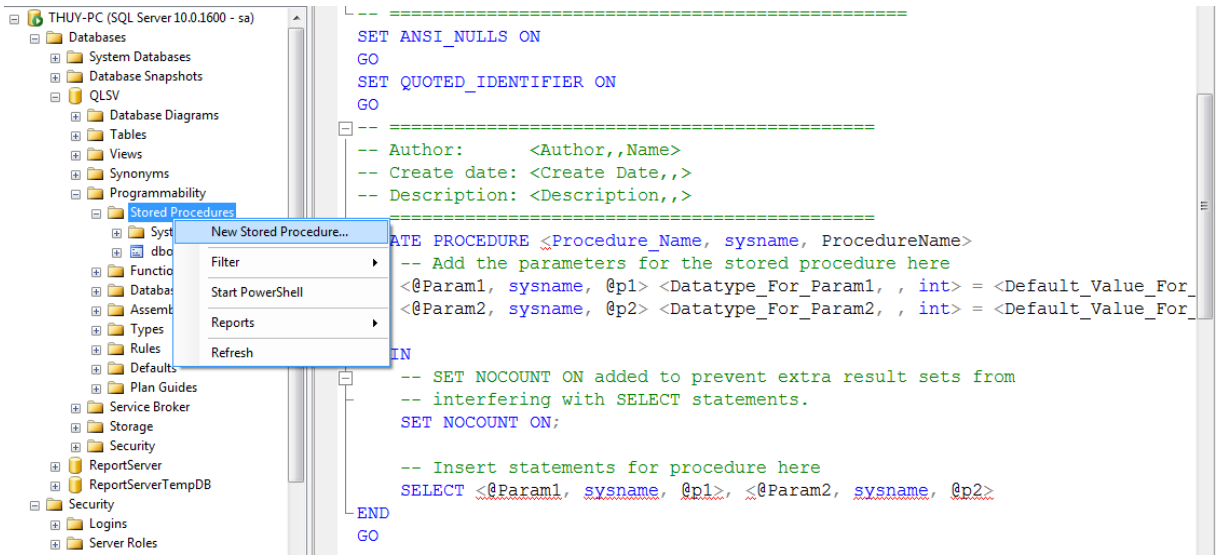
Trong SQL Server 2008, thủ tục được lưu trữ trong thư mục như sau:



3.5.1. Lệnh tạo thủ tục

3.5.1.1 Tạo bằng SQL Server Management Studio (SSMS):

Click chuột phải vào Stored Procedure → chọn New Stored Procedure, cửa sổ lệnh xuất hiện, ta chỉ cần thay đổi tên thủ tục, các tham số và nội dung của thủ tục.



3.5.1.2 Tạo bằng lệnh:

Cú pháp:

```
CREATE PROCEDURE Tên_thủ_tục [@ <tên tham số 1> <kiểu DL>
                                        [= <giá trị>] [OUTPUT], . . .]
[WITH RECOMPILE|ENCRYPTION|RECOMPILE, ENCRYPTION]
AS
    <Các_câu_lệnh_của_thủ_tục>
    [return [return_value] ]
```

Trong đó:

- Tên_thủ_tục: Là tên thủ tục được tạo mới, tên thủ tục này phải là duy nhất trong một CSDL, tuân theo quy tắc định danh.
- Danh_sách_tham_số: được khai báo ngay sau tên thủ tục, nếu thủ tục có nhiều tham số thì các tham số được ngăn cách nhau bởi dấu phẩy. Các tham số này được khai báo như sau: @<tên tham số> <kiểu dữ liệu> [(độ dài)] [output]
- WITH RECOMPILE: Thông thường thủ tục sẽ được biên dịch sẵn ở lần gọi đầu tiên. Nếu tùy chọn WITH RECOMPILE được chọn, SQL Server sẽ biên dịch lại thủ tục mỗi khi được gọi.
- WITH ENCRYPTION: Thủ tục sẽ được mã hóa nếu tùy chọn WITH ENCRYPTION được chọn. Nếu thủ tục đã được mã hóa, ta không thể xem được nội dung của thủ tục.
- Các_câu_lệnh_của_thủ_tục: Các lệnh T-SQL. Các lệnh này có thể nằm trong cặp BEGIN…END hoặc không.

Ví dụ 52: Thủ tục sau nhập vào họ tên và in ra ra câu 'Xin chào ' họ tên mà bạn nhập vào.

```
CREATE PROCEDURE XINCHAO
@HOTEN NVARCHAR(40), @LOP NVARCHAR(30)
AS
    BEGIN
        PRINT N'XIN CHÀO ' + @HOTEN + ' ' + @LOP
    END
GO
```

3.5.2. Tham số trong thủ tục

3.5.2.1. Tham số đầu vào

Tham số đầu vào là loại tham số mặc định, cho phép truyền các giá trị vào trong thủ tục để xử lý.

Cú pháp: khai báo tham số đầu vào như sau:

```
@<Tên tham số đầu vào>  <kiểu dữ liệu>[(độ dài)]
```

Ví dụ 53: Thủ tục sau đây in ra tổng của 2 số nguyên a và b. Thủ tục này cần cung cấp 2 tham số đầu vào là @a và @b.

```
CREATE PROC CONG @a int, @b int
As
    PRINT @a + @b
GO
```

Ví dụ 54: Xét bảng dữ liệu SINHVIEN như sau:

	Masv	Hoten	Ngsinh	Dchi	GioiTinh	Malop
1	SV01	Nguyen Thi Lan	1978-12-03 00:00:00	TPHCM	Nu	L001
2	SV02	Tran Thanh Tung	1980-02-04 00:00:00	VUNG TAU	Nam	L001
3	SV03	TRUNG THI HUE	1984-11-10 00:00:00	DA NANG	Nu	L001
4	SV04	LE VAN KHANH	1985-04-07 00:00:00	Vung Tau	Nam	L002
5	SV05	NGO VIET NAM	1986-08-14 00:00:00	Da Nang	Nam	L002
6	SV06	TRAN THI LIEU	1985-07-15 00:00:00	TPHCM	Nu	L003
7	SV07	Tran Thanh Nam	1988-02-04 00:00:00	Dong Nai	Nam	L004
8	SV08	Pham Hoai Phong	1979-05-19 00:00:00	Tien Giang	Nam	L004
9	SV09	Pham Mai Huon...	1981-05-19 00:00:00	VUNG TAU	Nu	L004

Tạo thủ tục có tên là DSLOP để xem danh sách sinh viên của 1 lớp nào đó. Thủ tục này cần cung cấp tham số đầu vào là @MALOP. Thủ tục được viết như sau:

```
CREATE PROC DSLOP @MALOP CHAR(10)
AS
    SELECT *
    FROM SINHVIEN
    WHERE Malop=@MALOP
GO
```

Ví dụ 55: Xét bảng dữ liệu KETQUA như sau:

MASV	MAMH	DIEM
SV001	MH001	7
SV002	MH001	9
SV002	MH002	5
SV003	MH003	8

Tạo thủ tục có tên là XEMDIEM để xem điểm một môn học nào đó của một sinh viên. Thủ tục này cần cung cấp hai tham số đầu vào là @MASV và @MAMH. Thủ tục được viết như sau:

```
CREATE PROC XEMDIEM @MASV CHAR(10), @MAMH CHAR(10)
AS
    SELECT DIEM
    FROM KETQUA
    WHERE MASV=@MASV AND MAMH=@MAMH
GO
```

3.5.2.2. Tham số đầu ra

Tham số đầu ra dùng để nhận kết quả mà thủ tục trả ra. Tham số đầu ra được sử dụng kết hợp với từ khóa OUTPUT

Trong các ví dụ trên chúng ta đã sử dụng các tham số để nhận giá trị truyền vào, còn được gọi là tham số đầu vào. Tuy nhiên trong thực tế đôi khi chúng ta cũng muốn nhận các giá trị trả về từ các xử lý bên trong thủ tục thông qua các tham số. Các tham số này được gọi là tham số đầu ra. Tham số đầu ra là những tham số mà giá trị của nó sẽ được tính toán từ bên trong thủ tục và sẽ được giữ nguyên sau khi thoát ra khỏi thủ tục. Tham số đầu ra được gắn với từ khoá OUTPUT trong lúc tạo thủ tục.

Ví dụ 56: Viết thủ tục nhập vào 2 số nguyên a và b, kết quả trả về là tổng 2 số a và b.

Thủ tục này cần cung cấp 2 tham số đầu vào là @a và @b, tham số @tong là tham số đầu ra để nhận kết quả trả về từ thủ tục. Thủ tục được viết như sau:

```
CREATE PROC CONG @a int, @b int, @tong int output
as
    Set @tong = @a + @b
GO
```

Ví dụ 57: Xét bảng dữ liệu SINHVIEN như sau:

	Masv	Hoten	Ngsinh	Dchi	GioiTinh	Malop
1	SV01	Nguyen Thi Lan	1978-12-03 00:00:00	TPHCM	Nu	L001
2	SV02	Tran Thanh Tung	1980-02-04 00:00:00	VUNG TAU	Nam	L001
3	SV03	TRUNG THI HUE	1984-11-10 00:00:00	DA NANG	Nu	L001
4	SV04	LE VAN KHANH	1985-04-07 00:00:00	Vung Tau	Nam	L002
5	SV05	NGO VIET NAM	1986-08-14 00:00:00	Da Nang	Nam	L002
6	SV06	TRAN THI LIEU	1985-07-15 00:00:00	TPHCM	Nu	L003
7	SV07	Tran Thanh Nam	1988-02-04 00:00:00	Dong Nai	Nam	L004
8	SV08	Pham Hoai Phong	1979-05-19 00:00:00	Tien Giang	Nam	L004
9	SV09	Pham Mai Huon...	1981-05-19 00:00:00	VUNG TAU	Nu	L004

Thủ tục sau đây sẽ trả về sĩ số của một lớp nào đó. Thủ tục sẽ có một tham số đầu vào là @Malop và một tham số đầu ra là @siso để nhận kết quả trả về từ thủ tục.

```
CREATE PROC SISOLOP
@MALOP CHAR(10), @SISO INT OUTPUT
AS
    SET @SISO = (SELECT COUNT(*)
                 FROM SINHVIEN
                 WHERE Malop = @MALOP)
GO
```

Ví dụ 58: Xét bảng dữ liệu KETQUA như sau:

MASV	**MAMH**	**DIEM**
SV001	MH001	7
SV002	MH001	9
SV002	MH002	5
SV003	MH003	8

Viết thủ tục sử dụng 2 tham số đầu vào là @MASV, @MAMH để nhận mã sinh viên và mã môn học truyền vào và một tham số đầu ra có tên là @DIEM để lấy giá trị điểm của sinh viên được xử lý trong thủ tục.

```
CREATE PROC XEMDIEM
@MASV CHAR(10), @MAMH CHAR(10),  @DIEM INT OUTPUT
AS
        SET @DIEM = (SELECT DIEM
                     FROM KETQUA
                     WHERE MASV=@MASV AND MAMH=@MAMH )
GO
```

3.5.3. Gọi thực thi thủ tục

Có 2 cách gọi thủ tục

Cú pháp 1: `Tên_thủ_tục[danh_sách_tham_số]`

Cú pháp này chỉ được sử dụng khi thủ tục được đặt ở vị trí đầu tiên trong một Batch SQL Script.

Cú pháp 2: `Exec[ute] Tên_thủ_tục[danh_sách_tham_số]`

Cú pháp này được sử dụng khi thủ tục được gọi từ một thủ tục khác, thực hiện bên trong một trigger hay phối hợp với các câu lệnh SQL khác.

Cách truyền tham số thực:

- Đối với tham số đầu vào: chỉ cần truyền tham số thực
- Đối với tham số đầu ra:
  - + Khai báo một biến có kiểu dữ liệu giống như tham số đầu ra
  - + Đặt biến này vào vị trí tham số đầu ra kèm theo từ khóa OUTPUT trong lời gọi thủ tục.
  - + Dùng lệnh Select hoặc lệnh Print biến vừa đặt để xem giá trị trả về của thủ tục.

Ví dụ 59: Giả sử thủ tục trả về sĩ số của một lớp được viết như sau:

```
CREATE PROC SISOLOP
@MALOP CHAR(10), @SISO INT OUTPUT
AS
    SET @SISO = (SELECT COUNT(*)
```

```
                    FROM SINHVIEN
                    WHERE Malop = @MALOP)
GO
```

Để biết được sĩ số của lớp ‘L001’, ta gọi thủ tục SISOLOP như sau:

```
DECLARE @SS INT
EXEC SISOLOP 'L001',@SS OUTPUT
PRINT @SS
```

Ví dụ 60: Giả sử thủ tục xem điểm một môn học của một sinh viên được viết như sau:

```
CREATE PROC XEMDIEM
@MASV CHAR(10), @MAMH CHAR(10),  @DIEM INT OUTPUT
AS
        SET @DIEM = (SELECT DIEM
                     FROM KETQUA
                     WHERE MASV=@MASV AND MAMH=@MAMH )
GO
```

Để xem điểm môn học ‘M001’ của sinh viên ‘SV01’, ta gọi thủ tục XEMDIEM như sau:

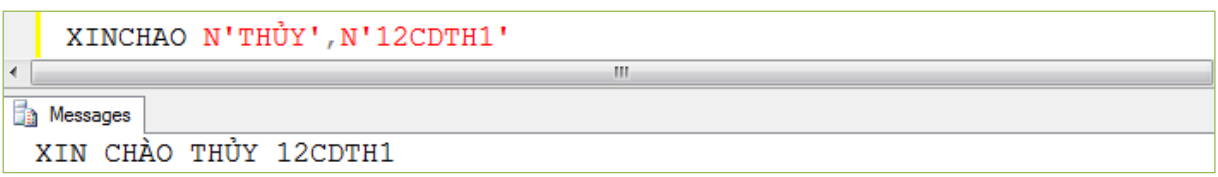
```
DECLARE @DIEM_SV INT
EXEC XEMDIEM 'SV01' , 'M001' , @DIEM_SV  OUTPUT
PRINT @DIEM_SV
```

Chú ý:

- Thông thường danh sách tham số thực truyền vào trong lời gọi phải theo đúng thứ tự khai báo các tham số hình thức trong thủ tục lưu trữ.
- Tuy nhiên, thứ tự của các tham số thực truyền vào cho thủ tục có thể không cần phải tuân theo thứ tự của các tham số hình thức như khi định nghĩa thủ tục nếu tất cả các tham số đều được viết dưới dạng:

```
@tên tham số = giá trị
```

Ví dụ 61: Để gọi thủ tục XINCHAO trên, ta thực hiện như sau:



Tuy nhiên, nếu đặt phía trước lời gọi thủ tục một câu lệnh khác thì sẽ báo lỗi như sau:

```
PRINT N'TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TPHCM'
XINCHAO N'THỦY', N'12CDTH1'
```

```
Messages
Msg 102, Level 15, State 1, Line 2
Incorrect syntax near 'XINCHAO'.
```

Cách xử lý lúc này là gọi thủ tục theo cú pháp 2, kết quả như sau:

```
PRINT N'TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TPHCM'
EXEC XINCHAO N'THỦY',N'12CDTH1'
```

```
Messages
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TPHCM
XIN CHÀO THỦY 12CDTH1
```

Trong trường hợp tham số thực truyền vào không đúng theo thứ tự đã khai báo trong thủ tục, ta gọi như sau:

```
PRINT N'TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TPHCM'
EXEC XINCHAO @LOP= N'12CDTH1',@HOTEN= N'THỦY'
```

```
Messages
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TPHCM
XIN CHÀO THỦY 12CDTH1
```

3.5.4. Sử dụng lệnh RETURN

Lệnh RETURN được sử dụng để trả về giá trị mà không cần sử dụng tham số đầu ra. Giá trị trả về này có một số đặc điểm sau:

- Giá trị trả về chỉ có thể là số nguyên. Nếu trả về các loại giá trị khác thì lúc thực thi thủ tục sẽ báo lỗi (ngoại trừ một số kiểu dữ liệu tự động chuyển sang kiểu số nguyên như float, double, …).
- Nếu lệnh RETURN không có giá trị chỉ định thì giá trị trả về mặc định là 0.
- Có thể nhận giá trị trả về này bằng 1 biến.
- Sau khi gọi RETURN, thủ tục sẽ trả về giá trị và thoát khỏi thủ tục.

Để gọi thủ tục trả về giá trị bằng lệnh RETURN, ta phải khai báo một biến để nhận giá trị trả về này theo cú pháp sau:

```
DECLARE <biến nhận trả về>
EXEC[UTE] <biến nhận trả về> = <tên thủ tục>[<tham số>]
```

Ví dụ 62: Thủ tục kiểm tra hiệu của 2 số A và B được viết như dưới đây:

```
CREATE PROC KTSO
@A INT, @B INT
AS
BEGIN
    IF (@A-@B) = 1
        RETURN 1
    IF (@A-@B) > 10
         BEGIN
            DECLARE @C FLOAT
            SET @C=(@A-@B)/3
            RETURN @C
         END
    IF (@A-@B) < 0
        RETURN N'A nhỏ hơn B'
END
GO
```

--Gọi thực hiện thủ tục như sau:

```
DECLARE @KQ FLOAT
EXEC @KQ= KTSO 24,4
PRINT @KQ
GO
```

Giải thích:

- ✓ Nếu A - B = 1: trả về giá trị 1
- ✓ Nếu A - B > 10: trả về giá trị (A-B)/3, giá trị này là số thực, được tự động chuyển thành kiểu số nguyên.
- ✓ Nếu A - B < 0:báo lỗi không thể chuyển chuỗi `'A nhỏ hơn B'` thành số nguyên.

✓ Nếu A – B = các giá trị khác, trả về giá trị 0.

Ví dụ 63: Xét SINHVIEN(MASV, HOTEN, NGSINH, GTINH, DIACHI, MALOP)

Thủ tục sau đây sử dụng lệnh RETURN để trả về tuổi của một sinh viên với tham số đầu vào là @MASV

```
CREATE PROC TINHTUOI @MASV CHAR(10)
AS
  DECLARE @TUOI INT
  SET @TUOI= (SELECT DATEDIFF(YY,NGSINH,GETDATE())
              FROM SINHVIEN
              WHERE MASV=@MASV)
  RETURN @TUOI
GO
--Gọi thực hiện thủ tục như sau:
DECLARE @TUOI INT
EXEC @TUOI= TINHTUOI 'SV02'
PRINT @TUOI
GO
```

3.5.5. Sử dụng giá trị mặc định cho tham số

Các tham số được khai báo trong thủ tục có thể nhận các giá trị mặc định. Giá trị mặc định sẽ được gán cho tham số trong trường hợp không truyền tham số thực cho tham số khi có lời gọi đến thủ tục. Tham số với giá trị mặc định được khai báo theo cú pháp như sau:

```
@<tên_tham_số> <kiểu_dữ_liệu> = <giá_trị_mặc_định>
```

Ví dụ 64: Cho MONHOC(MAMH, TENMH, SOTC)

Thủ tục sau thêm 1 môn học mới với 3 tham số đầu vào là @MAMH, @TENMH, @SOTC. Trong đó @SOTC có giá trị mặc định là 2. Trước khi thêm mới, thủ tục kiểm tra nếu môn học đó đã tồn tại rồi thì xuất ra câu thông báo : 'Môn học đã tồn tại'.

```
CREATE PROC THEM_MONHOC
@MAMH CHAR(5), @TENMH NVARCHAR(30), @SOTC INT = 2
AS
    BEGIN
```

```
    IF EXISTS (SELECT * FROM MONHOC WHERE Mamh=@MAMH)
        BEGIN
         PRINT N'MÔN HỌC ' + @MAMH + N'ĐÃ TỒN TẠI'
         RETURN -1
        END
    INSERT INTO MONHOC
    VALUES(@MAMH, @TENMH, @SOTC)
    END
GO
```

-- Gọi thực hiện thủ tục, trường hợp truyền đầy đủ tham số: môn học HE QTCSDL sẽ được thêm vào với số tín chỉ bằng 3

```
EXEC THEM_MONHOC 'M014','HE QTCSDL',3
```

-- Gọi thực hiện thủ tục, trường hợp không truyền tham số mặc định: môn học CONG NGHE WEB sẽ được thêm vào với số tín chỉ mặc định bằng 2

```
EXEC THEM_MONHOC 'M015','CONG NGHE WEB’
```

Ví dụ 65: Viết thủ tục in ra thông tin sinh viên khi truyền vào tham số @MASV có giá trị mặc định là ‘SV01’

```
CREATE PROC TT_SV @MASV CHAR(10)='SV01'
AS
    SELECT *
    FROM SINHVIEN
    WHERE MASV=@MASV
GO
```

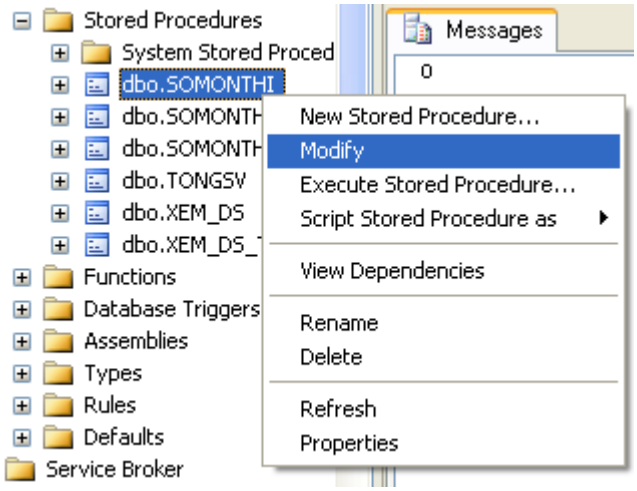
Khi gọi thực hiện thủ tục mà không cung cấp mã sinh viên thì giá trị mặc định sẽ được sử dụng là ‘SV01’

3.5.6. Quản lý thủ tục

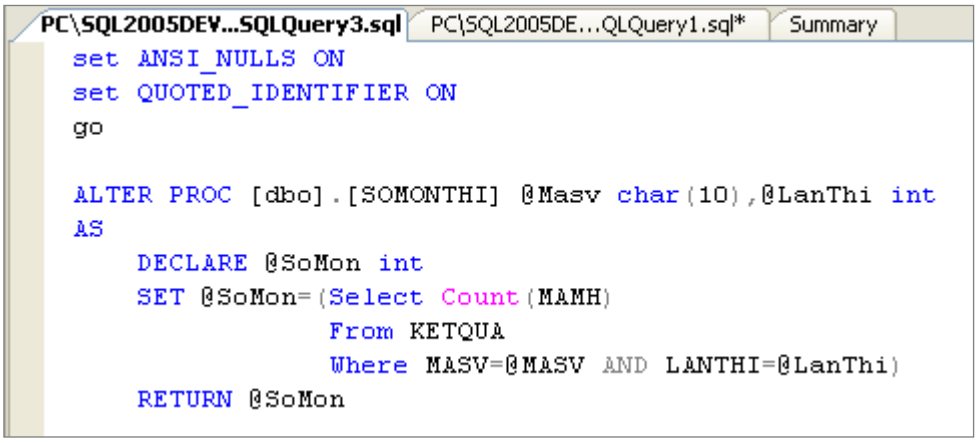
3.5.6.1. Sửa nội dung thủ tục

Để chỉnh sửa nội dung một thủ tục đã tồn tại, thực hiện như sau:

Từ danh mục các thủ tục Click chuột phải vào thủ tục cần chỉnh sửa → chọn Modify



Khi đó một cửa sổ lệnh hiện ra, trên cửa sổ này ta tiến hành sửa lại nội dung thủ tục.

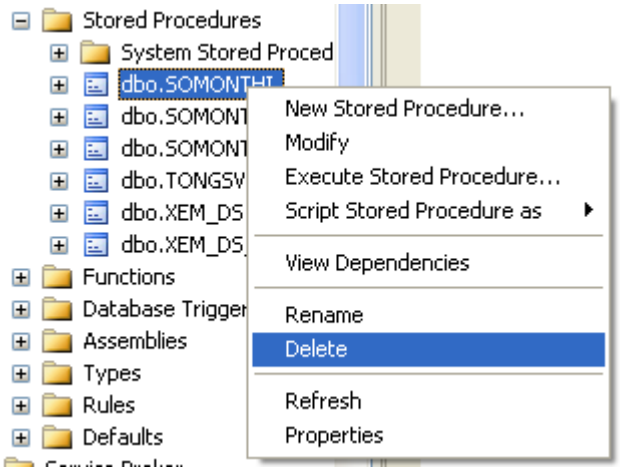


Sau khi sửa song thì bôi đen chọn khối lệnh thủ tục rồi thực thi thì thủ tục sẽ được chỉnh sửa.

✍ **Lưu ý:** Lệnh ALTER PROC trong cửa sổ lệnh sửa thủ tục ở trên có chức năng sửa thủ tục. Do đó, chúng ta có thể sử dụng lệnh này trực tiếp trước tên thử tục mà không cần chọn modify như đã trình bày.

3.5.6.2. Xoá thủ tục

Những thủ tục không còn sử dụng nữa thì ta có thể xoá đi bằng cách: Click chuột phải vào tên thủ tục và chọn Delete → chọn OK.



Ngoài ra, có thể xoá thủ tục bằng lệnh theo cấu trúc:

```
DROP PROC[EDURE] <tên thủ tục 1>,<tên thủ tục 2>,...
```

Ví dụ 66: Xoá 2 thủ tục có tên là TONGSV và XEM_DS, thực hiện như sau:

```
DROP PROC TONGSV,XEM_DS
```

3.6. Hàm do người dùng định nghĩa (Function)

Hàm là đối tượng cơ sở dữ liệu tương tự như thủ tục. Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm còn thủ tục thì không. Ngoài những hàm do hệ quản trị cơ sở dữ liệu cung cấp sẵn, người sử dụng tự định nghĩa thêm các hàm nhằm phục vụ cho mục đích riêng của mình.

3.6.1. Hàm vô hướng (Scalar – Valued Function)

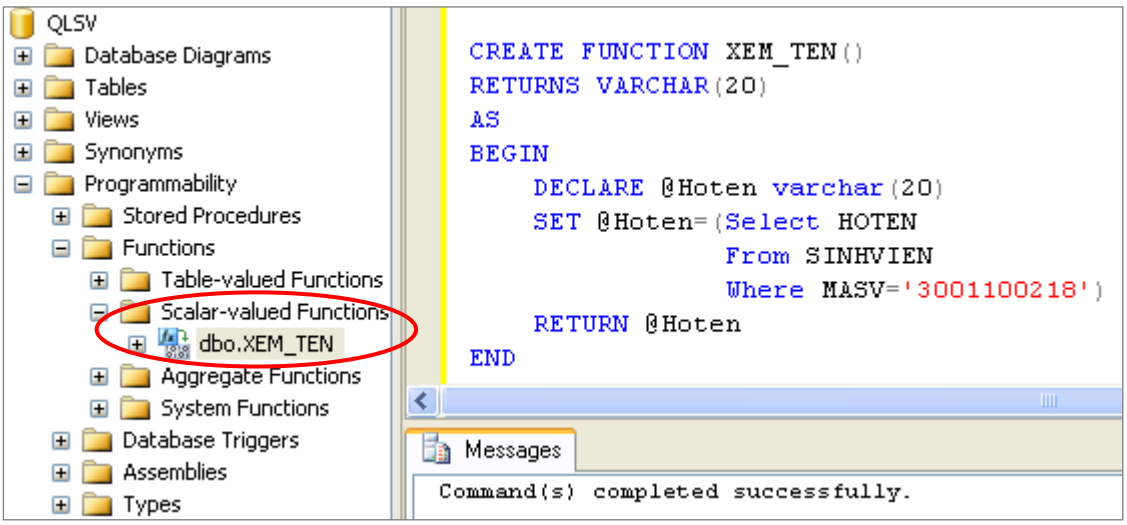
Hàm vô hướng là hàm sẽ trả về duy nhất một giá trị vô hướng bằng lệnh RETURN, các giá trị trả về có thể là số, chuỗi, ngày tháng, ...

Trong SQL Server, các hàm dạng này lưu trong mục Scalar-Valued Functions.

Cú pháp:

```
CREATE FUNCTION <Tên hàm>([<tham số1>,<tham số2>,...])
RETURNS <kiểu dữ liệu trả về của hàm>
AS
    BEGIN
        <Các lệnh T-SQL>
    END
```

Ví dụ 67: Tạo hàm không tham số, trả về họ tên của sinh viên có mã là: '3001100218'

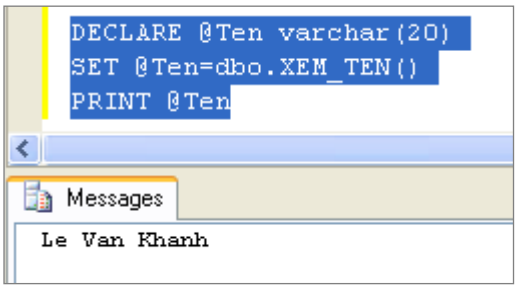


Thực hiện tạo hàm như sau: Gõ lệnh tạo hàm vào cửa sổ lệnh → chọn khối đoạn lệnh tạo hàm vừa gõ → nhấn nút thực thi trên thanh công cụ (F5) để thực thi lệnh tạo hàm.

Sau khi tạo ra hàm, ta gọi hàm theo cú pháp sau:

```
DECLARE <biến nhận trả về>
SET < biến nhận trả về> = dbo.<tên hàm>(danh sách
                          giá trị truyền cho tham số)
```

Trong ví dụ trên, hàm XEM_TEN được gọi như sau:

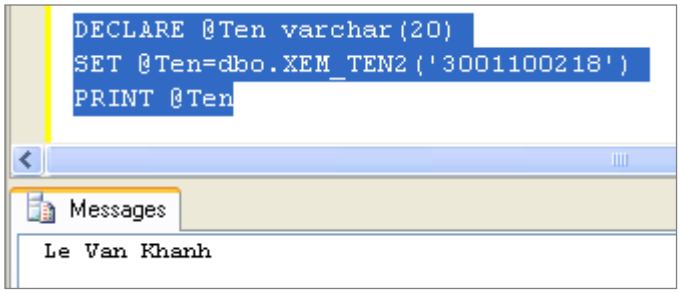


Ví dụ 68: Tạo hàm truyền vào tham số là mã sinh viên sẽ trả về họ tên của sinh viên đó.

```
CREATE FUNCTION XEM_TEN2(@Masv char(10))
RETURNS VARCHAR(20)
AS
    BEGIN
        DECLARE @Hoten varchar(20)
        SET @Hoten=(Select HOTEN From SINHVIEN
```

```
                    Where MASV=@Masv)
        RETURN @Hoten
END
```

Trong ví dụ trên, hàm XEM_TEN2 được gọi như sau:



3.6.2. Hàm nội tuyến (Table – Valued Function)

Hàm nội tuyến là hàm trả về giá trị là một bảng dữ liệu. Loại hàm này cung cấp chức năng như khung nhìn (View) nhưng linh hoạt hơn nhờ cho phép sử dụng các tham số.

Cú pháp 1: hàm trả về dữ liệu được lấy từ các bảng trong cơ sở dữ liệu, giống như View nhưng có tham số đầu vào.

```
CREATE FUNCTION <tên_hàm>([danh sách các tham số])
RETURNS TABLE
AS
        RETURN(câu lệnh Select)
```

Quy tắc áp dụng cú pháp 1:

– Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

Cách gọi hàm:

```
SELECT tên_cột FROM dbo.tên_hàm(các đối số)
```

Ví dụ 69: Cho SINHVIEN(MASV, HOTEN, NGSINH, DCHI, GTINH, MALOP)

Ta định nghĩa hàm XEM_DSLOP trả về danh sách sinh viên của một lớp bất kỳ như sau:

```
CREATE FUNCTION XEM_DSLOP (@MALOP CHAR(10))
RETURNS TABLE
AS
```

```
    RETURN(SELECT * FROM SINHVIEN
            WHERE MALOP = @MALOP)
GO
```

--Gọi hàm

```
SELECT * FROM dbo.XEM_DSLOP('12CDTH1')
GO
```

Hàm trên nhận tham số đầu vào là mã lớp của sinh viên cần xem và giá trị trả về là tập hợp các dòng dữ liệu cho biết thông tin về các sinh viên của lớp đó.

Ví dụ 70: Cho SINHVIEN(MASV, HOTEN, NGSINH, DCHI, GTINH, MALOP)

Hàm sau trả về sĩ số của tất cả các lớp, hàm này không có tham số đầu vào, giá trị trả về là một table gồm 2 cột mã lớp và sĩ số.

```
CREATE FUNCTION DS_SISO()
RETURNS TABLE
AS
    RETURN(SELECT MALOP, COUNT(*) AS SISO
            FROM SINHVIEN
            GROUP BY MALOP )
GO
```

--Gọi hàm

```
select * from dbo.DS_SISO()
```

✍ **Lưu ý**: Đối với cú pháp trên, phần thân của hàm chỉ cho phép sử dụng một câu lệnh Select duy nhất. Trong trường hợp phải sử dụng nhiều câu lệnh khác trong phần thân của hàm thì cú pháp trên không sử dụng được. Khi đó, ta áp dụng cú pháp 2 sau đây:

Cú pháp 2: Tạo và trả về một bảng

```
CREATE FUNCTION<tên_hàm>([danh sách các tham số])
RETURNS @Tên_bảng_trả_về TABLE (Tên_cột KDL[,…])
AS
     BEGIN
        --Các câu lệnh trong thân hàm
        Insert into @tên_bảng_trả_về ..
        RETURN
     END
```

Quy tắc áp dụng cú pháp 2:

- Cấu trúc của bảng trả về được xác định dựa vào định nghĩa của bảng trong mệnh đề RETURNS. Biến @Tên_bảng_trả_về có phạm vi sử dụng trong hàm và được sử dụng như một tên bảng.
- Câu lệnh RETURN trong thân hàm không chỉ định giá trị trả về. Giá trị trả về của hàm chính là các dòng dữ liệu của bảng được định nghĩa trong mệnh đề RETURNS.

Ví dụ 71: Cho MONHOC(MAMH, TENMH, SOTIET)

Hàm sau đây nhập vào một số nguyên, nếu nhập số 1 thì trả về các môn học có số tiết <=30, nếu nhập số 2 thì trả về các môn học có 30 <số tiết <=60. Nếu nhập các số khác thì trả về các môn học có số tiết > 60.

```
CREATE FUNCTION DS_MONHOC(@SO INT)
RETURNS @DS TABLE
                ( MAMH CHAR(10),
                  TENMH NVARCHAR(30),
                  SOTIET INT
                 )
AS
    BEGIN
        IF @SO = 1
            INSERT INTO @DS
                SELECT *
                FROM MONHOC
                WHERE Sotiet <=30
        ELSE IF @SO = 2
            INSERT INTO @DS
            SELECT *
            FROM MONHOC
            WHERE Sotiet > 30 AND Sotiet <=60
        ELSE
            INSERT INTO @DS
            SELECT *
```

```
            FROM MONHOC
            WHERE Sotiet > 60
    RETURN
END
GO
```

3.7. Trigger

Trong một cơ sở dữ liệu, việc đảm bảo các ràng buộc toàn vẹn là vô cùng quan trọng. Ở các chương trước, chúng ta đã được học các loại ràng buộc như primary key constraint, foreign key constraint, check constraint, unique constraint, default constraint, rule … Các loại ràng buộc này phần nào đáp ứng được nhu cầu kiểm tra ràng buộc toàn vẹn trong một cơ sở dữ liệu. Tuy nhiên, đối với các ràng buộc phức tạp, khi mà các loại constraint trên không thể đáp ứng thì trigger là sự lựa chọn mà chúng ta nghĩ tới.

3.7.1. Khái niệm trigger

Trigger là một dạng đặc biệt của thủ tục lưu trữ có khả năng thực thi một cách tự động mỗi khi dữ liệu trên bảng liên quan với trigger được cập nhật (thêm, xoá, sửa). Có nghĩa là mỗi một trigger được tạo ra sẽ gắn liền với một bảng nào đó trong cơ sở dữ liệu. Khi dữ liệu trong bảng bị thay đổi thì trigger sẽ được tự động kích hoạt.

Trigger được chia ra làm hai nhóm: DML và DDL trigger

- DML trigger thực thi khi người sử dụng cố gắng sửa đổi dữ liệu thông qua các sự kiện thao tác dữ liệu (data manipulation language) như insert, update, delete trên table hoặc view.
- DDL trigger thực thi đáp ứng các sự kiện định nghĩa dữ liệu (data definition language).

Trong phạm vi giáo trình này, tác giả chỉ trình bày về DML trigger.

✍**Lưu ý: Các trường hợp cần sử dụng Trigger:**

- Khi các loại constraint không thể thỏa mãn yêu cầu của ứng dụng. Các loại constraint thuộc loại Declarative Data Integrity cho nên sẽ kiểm tra dữ liệu trước khi cho phép nhập vào table, còn trigger thuộc loại Procedural Data Integrity nên các sự kiện insert, update, delete đã xảy ra rồi mới kích hoạt trigger.
- Khi một cơ sở dữ liệu chưa được chuẩn hóa, sẽ có một số dữ liệu được chứa cùng lúc trong nhiều table. Khi đó, để đảm bảo tính thống nhất về mặt dữ liệu, ta sử dụng trigger, có nghĩa là khi dữ liệu cập nhật ở table này thì cũng tự động cập nhật ở table khác.

– Khi thay đổi dây chuyền dữ liệu giữa các bảng với nhau (dữ liệu của bảng này thay đổi thì dữ liệu của bảng khác cũng bị thay đổi theo).

Đặc điểm của trigger:

– Trigger không có tham số

– Một trigger có thể nhận biết, ngăn chặn và hủy bỏ những thao tác làm thay đổi trái phép dữ liệu trong cơ sở dữ liệu.

– Một trigger có thể làm nhiều công việc khác nhau và có thể được kích hoạt bởi nhiều hơn một sự kiện.

– Trigger không thể được tạo ra trên temporary hay system table.

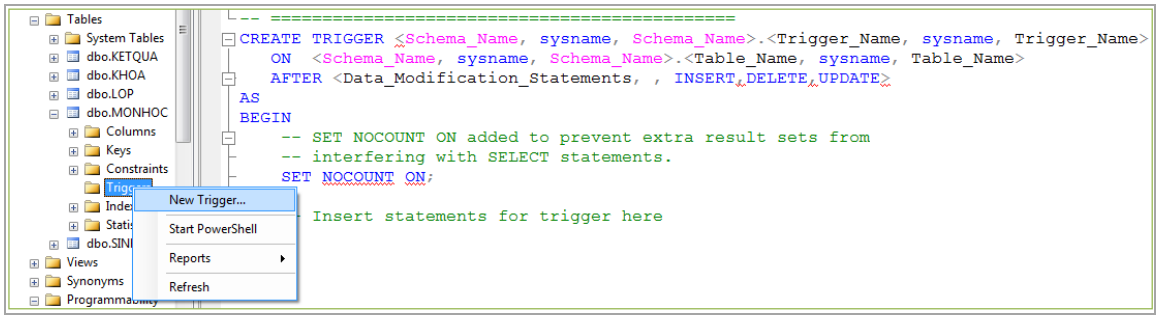
– Trigger chỉ có thể được kích hoạt một cách tự động bởi một trong các sự kiện insert, update, delete mà không thể chạy manualy được.

– Có thể áp dụng trigger cho View.

3.7.2. Tạo trigger

3.7.1.1. Tạo trigger bằng công cụ Management Studio

Chọn bảng cần tạo trigger → click phải vào Trigger → chọn New Trigger → tinh chỉnh lại đoạn lệnh tạo trigger theo từng ràng buộc.



3.7.1.2. Tạo trigger bằng lệnh

```
CREATE TRIGGER <tên trigger>
ON <tên bảng>/<tên View>
[WITH ENCRYPTION]
FOR|AFTER|INSTEAD OF  INSERT| DELETE|UPDATE
AS
   <Các lệnh T-SQL>
```

Trong đó:

– Tên trigger: tên của trigger cần tạo, tên này phải duy nhất trong một CSDL.

- Từ khoá ON: Chỉ ra rằng trigger này đang viết cho bảng dữ liệu nào.
- Tên bảng/tên View: Là tên của bảng dữ liệu hay View cần tạo trigger trên đó.
- Mệnh đề WITH ENCRYPTION: Dùng để mã hoá nội dung trigger.
- FOR|AFTER|INSTEAD OF: loại trigger
- INSERT, DELETE, UPDATE: Hành động kích hoạt trigger
- Các lệnh T-SQL: là các lệnh SQL thực hiện các xử lý trong trigger.

3.7.3. Phân loại trigger

DML trigger chia thành 2 loại là AFTER trigger và INSTEAD OF trigger. Trong đó AFTER trigger là kiểu phổ biến nhất.

3.7.3.1. AFTER trigger

Khi câu lệnh insert/update/delete được thực hiện trên một bảng có định nghĩa trigger, đầu tiên SQL sẽ kiểm tra các ràng buộc, nếu vi phạm một trong các ràng buộc đã định nghĩa trước đó thì sẽ dừng câu lệnh trên và các trigger không được kích hoạt. Ngược lại, nếu thành công thì các trigger sẽ được thực thi. Có nghĩa là trigger được thực thi sau khi các thao tác insert/update/delete đã thực hiện thành công.

After trigger là trigger mặc định nếu chỉ có chỉ định FOR.

a. Insert trigger

Khi hành động thêm (INSERT) dữ liệu vào bảng xảy ra thì Insert trigger trên bảng này sẽ được kích hoạt. Khi đó hệ thống sẽ phát sinh ra một bảng phụ có tên là INSERTED để chứa dòng dữ liệu mới mà người dùng thêm vào bảng, bảng phụ này có cấu trúc hoàn toàn giống với bảng được định nghĩa trigger trên đó. Sau khi trigger kết thúc hoạt động thì bảng phụ này sẽ bị huỷ bỏ.

Ví dụ 72: Cho MONHOC(MAMH, TENMH, SOTC)

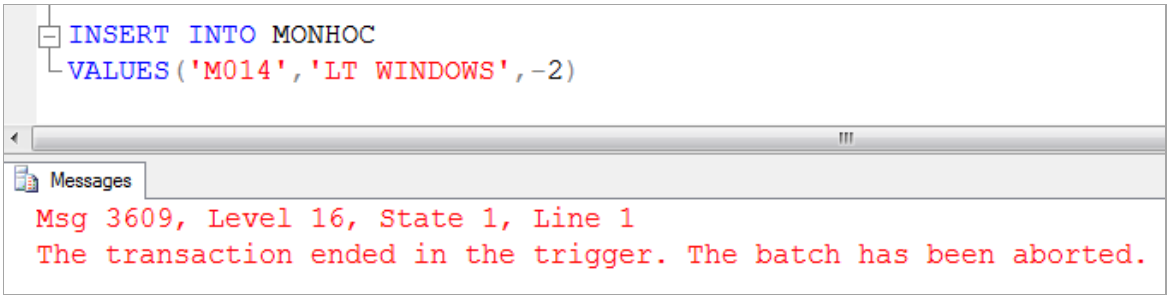
Tạo trigger kiểm tra khi nhập dữ liệu vào bảng MONHOC thì SOTC phải > 0

```
CREATE TRIGGER KT_SOTIET
ON MONHOC
FOR INSERT
AS
IF (SELECT SOTC FROM INSERTED) < 0
   ROLLBACK TRAN
GO
```

Trong ví dụ trên chúng ta sử dụng mệnh đề For Insert để chỉ ra rằng trigger này chỉ được kích hoạt khi thực hiện hành động insert dữ liệu vào bảng MONHOC.

Lệnh `IF(SELECT SOTC FROM INSERTED)<0` sẽ kiểm tra nếu số tín chỉ trong bảng tạm (Inserted) < 0 nghĩa là dữ liệu không hợp lệ thì thực hiện quay lui giao tác bởi lệnh `ROLLBACK TRAN`. Khi đó, dữ liệu sẽ không được chèn vào bảng.

Khi thêm 1 dòng dữ liệu vào bảng MONHOC với số tín chỉ không hợp lệ thì sẽ xuất hiện lỗi như hình dưới đây:



b. Delete trigger

Khi hành động xoá (DELETE) dữ liệu trên bảng xảy ra thì Delete Trigger trên bảng đó sẽ được kích hoạt. Khi đó một bảng phụ có tên là Deleted phát sinh để chứa dòng dữ liệu mà người sử dụng đã xoá đi. Bảng phụ này cũng có cấu trúc giống với bảng được định nghĩa trigger trên đó và sẽ bị huỷ bỏ sau khi kết thúc hoạt động của trigger.

Ví dụ 73: Cho bảng GIAODICH(MAGD, MAKH, NGAYGD) lưu trữ thông tin khách hàng giao dịch với ngân hàng.

Cần viết trigger kiểm tra chỉ cho xoá những khách hàng có thời gian giao dịch trước năm 2008

```
CREATE TRIGGER Xoa_KH ON GIAODICH
FOR DELETE
AS
    If (SELECT year(NGAYGD) FROM DELETED)>= 2008
        ROLLBACK TRAN
GO
```

Trong ví dụ trên chúng ta sử dụng mệnh đề For Delete để chỉ ra rằng trigger này chỉ được kích hoạt khi thực hiện hành động delete dữ liệu ra khỏi bảng GIAODICH.

Lệnh `If(SELECT year(NGAYGD) FROM DELETED)>=2008` sẽ kiểm tra nếu năm giao dịch từ 2008 trở về sau có nghĩa là dữ liệu không hợp lệ thì thực hiện quay lui giao tác bởi lệnh `ROLLBACK TRAN`. Khi đó, dữ liệu sẽ không bị xóa khỏi bảng.

c. Update trigger

Khi hành động sửa đổi (UPDATE) dữ liệu trên một bảng xảy ra thì Update Trigger trên bảng đó sẽ được kích hoạt, khi đó hệ thống sẽ phát sinh ra 2 bảng phụ: một bảng dùng để chứa dữ liệu mới cần sửa vào có tên là Inserted, bảng còn lại dùng để chứa dữ liệu cũ cần bỏ đi có tên là Deleted. (Hành động sửa dữ liệu tương ứng với 2 hành động: xoá dữ liệu cũ sau đó thêm dữ liệu mới). Hai bảng phụ này cũng bị huỷ bỏ sau khi kết thúc hoạt động của trigger.

Ví dụ 74: Cho bảng HANG(MAHG, TENHG, DONGIA)

Trigger sau đây chỉ cho phép cập nhật tăng đơn giá mặt hàng không quá 10% so với đơn giá cũ.

```
CREATE TRIGGER KT_DONGIA ON HANG
FOR UPDATE
AS
    IF (SELECT DONGIA From INSERTED)>1.1*(SELECT
                                        DONGIA FROM DELETED)
    ROLLBACK TRAN
GO
```

d. Trigger nhiều hành động

Một trigger khi được tạo có thể khai báo cho nhiều hành động khác nhau: insert, delete, update thay vì phải viết 3 trigger tương ứng với mỗi hành động. Trigger dạng này sẽ có không gian hoạt động rộng hơn và hiệu quả hơn.

Trở lại ví dụ như trong phần Insert Trigger thêm vào một hành động kích hoạt là Update khi đó trigger trở thành:

```
CREATE TRIGGER KT_SOTIET
ON MONHOC
FOR INSERT, UPDATE
AS
    IF (SELECT SOTC FROM INSERTED) < 0
        ROLLBACK TRAN
GO
```

Như vậy trigger này sẽ được kích hoạt với một trong 2 hành động là insert và update.

3.7.3.2. INSTEAD OF trigger

- Instead of trigger được thực thi thay cho thao tác insert/update/delete tương ứng. Có nghĩa là nó được kích hoạt trước khi bất cứ một dữ liệu nào được thay đổi trong cơ sở dữ liệu.
- Các ràng buộc không được kiểm tra trước khi trigger kích hoạt.
- Các bảng tạm inserted và deleted vẫn được tạo ra.
- Thường được dùng để xử lý cập nhật trên View, rất hữu ích trong trường hợp cập nhật nhiều bảng một lúc trong khung nhìn.

Ví dụ 75: Cho LOP(MALOP, TENLOP)

SINHVIEN(MASV, HOTEN, MALOP)

Xét trigger Instead Of được viết dưới đây:

```
CREATE TRIGGER VD1 ON SINHVIEN
INSTEAD OF INSERT
AS
    DECLARE @MALOP CHAR(10)
    SET @MALOP=(SELECT MALOP FROM INSERTED)
    IF NOT EXISTS(SELECT * FROM LOP
                  WHERE MALOP=@MALOP)
        INSERT INTO LOP
        VALUES(@MALOP,'CHUA BIET')
GO
-------------------------------------------
INSERT INTO SINHVIEN
VALUES ('SVA','NGUYEN VAN A','L100')
```

Sau khi thực hiện câu lệnh thêm vào sinh viên `'NGUYEN VAN A'` với mã lớp `'L100'` chưa có trong bảng lớp, kết quả như sau:

- Bảng SINHVIEN không được chèn dữ liệu do trigger đã được thực thi thay cho câu lệnh `INSERT INTO SINHVIEN` ban hành.
- Bảng LOP được thêm 1 dòng mới là `'L100', 'CHUA BIET'` do thực thi nội dung câu lệnh bên trong trigger.

Ví dụ 76: View và Instead Of trigger

Cho LOP(MALOP, TENLOP)

SINHVIEN(MASV, HOTEN, MALOP)

Tạo View SV_LOP lấy thông tin từ 2 bảng SINHVIEN và LOP như sau:

```
CREATE VIEW SV_LOP
AS
    SELECT MASV, HOTEN, S.MALOP, TENLOP
    FROM SINHVIEN S, LOP L
    WHERE S.Malop = L.MALOP
GO
```

Ta muốn thêm 1 dòng vào view, sử dụng câu lệnh sau:

```
INSERT INTO SV_LOP
VALUES('SV11','NGUYEN VAN A','L100','12CDTH1')
```

Hệ thống sẽ báo lỗi không thể cập nhật View do có nhiều bảng trong mệnh đề From của câu lệnh Select tạo View đó.

```
View or function 'SV_LOP' is not updatable because the
modification affects multiple base tables.
```

Để giải quyết vấn đề trên, thay vì thêm dòng trên vào view SV_LOP, ta thêm dòng `('L100','12CDTH1')` vào table LOP và dòng `('SV11','NGUYEN VAN A','L100')` vào table SINHVIEN. Trigger Instead Of giúp ta làm điều này như sau:

```
CREATE TRIGGER VD2 ON SV_LOP
INSTEAD OF INSERT
AS
IF NOT EXISTS(SELECT * FROM LOP
            WHERE MALOP=(SELECT MALOP FROM INSERTED))
               AND NOT EXISTS(SELECT * FROM SINHVIEN
                           WHERE MASV=(SELECT MASV
                                  FROM INSERTED))
    BEGIN
        INSERT INTO LOP
            SELECT MALOP, TENLOP FROM INSERTED
```

```
        INSERT INTO SINHVIEN
            SELECT MASV, HOTEN, MALOP FROM INSERTED
    END
    ELSE
        PRINT N'SINH VIÊN HOẶC LỚP ĐÃ TỒN TẠI'
GO
```

Trigger ở ví dụ trên kiểm tra nếu mã lớp và mã sinh viên chưa tồn tại thì thêm mới một dòng vào bảng LOP và một dòng vào bảng SINHVIEN. Ngược lại, xuất ra câu thông báo 'SINH VIÊN HOẶC LỚP ĐÃ TỒN TẠI'.

3.7.4. Sử dụng IF UPDATE trong trigger

Thay vì chỉ định một trigger kích hoạt trên một bảng, ta có thể chỉ định trigger kích hoạt và thực hiện những thao tác cụ thể khi việc thay đổi dữ liệu chỉ liên quan đến một số cột nhất định nào đó trên bảng. Lúc này, ta sử dụng mệnh đề IF UPDATE trong trigger. Mệnh đề IF UPDATE có thể xuất hiện nhiều lần trong phần thân của trigger. Khi đó, mệnh đề nào đúng thì phần câu lệnh trong mệnh đề đó sẽ được thực thi.

Ví dụ 77: Cho LOP(MALOP, TENLOP, SISO)

SINHVIEN(MASV, HOTEN, NGSINH, GTINH, DCHI, MALOP)

Trigger sau đây cập nhật lại sĩ số của một lớp khi ta thay đổi mã lớp của một sinh viên.

```
CREATE TRIGGER SISO_UPDATE
ON SINHVIEN
FOR UPDATE
AS
    IF UPDATE(MALOP)
    BEGIN
        UPDATE LOP
        SET SISO=SISO+1
        WHERE MALOP= (SELECT MALOP FROM INSERTED)

        UPDATE LOP
        SET SISO=SISO-1
        WHERE MALOP= (SELECT MALOP FROM DELETED)
```

```
    END
GO
```

Trong ví dụ trên, khi ta thực hiện câu lệnh:

```
UPDATE SINHVIEN
SET MALOP = 'L001'
WHERE MASV ='SV04'
```

Hệ thống sẽ kích hoạt mệnh đề `IF UPDATE(MALOP)` trong trigger trên và câu lệnh update trong trigger sẽ được thực thi. Tuy nhiên, nếu ta thực hiện câu lệnh sau thì trigger này sẽ không được kích hoạt:

```
UPDATE SINHVIEN
SET GTINH = N'Nữ'
WHERE MASV ='SV04'
```

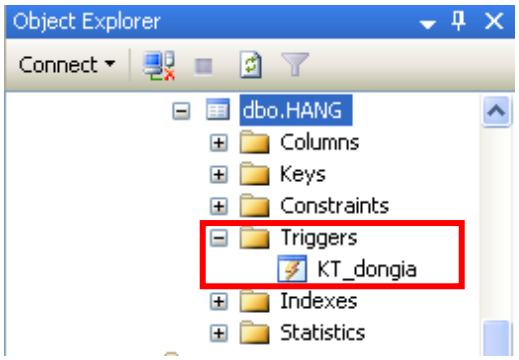
✍ **Lưu ý:**

- Thứ tự kiểm tra các ràng buộc toàn vẹn dữ liệu trong bảng : Trigger Instead Of → Constraint → Trigger For|After
- Lệnh tạo trigger phải là lệnh đầu tiên trong một query batch.
- Không thể định nghĩa trigger AFTER cho View.
- Trên một bảng có thể định nghĩa *nhiều* trigger For|After cho mỗi thao tác nhưng chỉ có thể định nghĩa *một* trigger Instead Of cho mỗi thao tác.
- Không thể định nghĩa trigger Instead Of Update/Delete trên bảng có cài đặt khóa ngoại dạng update cascade/ delete cascade.
- Một số lệnh sau đây không được sử dụng trong trigger: ALTER DATABASSE, CREATE DATABASE, DISK INIT, DISK RESIZE, DROP DATABASE, LOAD DATABASE, LOAD LOG, RECONFIGURE, RESTORE DATABASE, RESTORE LOG.
- Trong trường hợp có nhiều trigger cùng liên quan đến một đối tượng, ta có thể xác định thứ tự thực hiện của chúng bằng thủ tục hệ thống sp_settriggerorder.

3.7.5. Quản lý trigger

Sau khi tạo các trigger trên các bảng dữ liệu trong cơ sở dữ liệu thì công việc tiếp theo là phải quản lý các trigger này như thế nào cho có hiệu quả.

- **Nơi lưu trữ trigger:** trigger được lưu trữ trực tiếp trên bảng, view. Tất cả các trigger có trên bảng và View được lưu trữ trong mục Triggers như hình sau:



- **Xem mã trigger:**

Dùng thủ tục hệ thống: `sp_helptext <tên bảng>`

Ví dụ 78: Xem nội dung trigger có tên KT_dongia

```
sp_helptext KT_dongia
```

Results | Messages

	Text
1	Create Trigger KT_dongia on HANG
2	For Update
3	AS
4	If(Select DONGIA From Inserted)<=1.1*(Select DONGIA From Deleted)
5	Commit tran
6	Else
7	Begin
8	Print 'Don gia khong hop le'
9	Rollback tran
10	End

- **Xem những trigger nào đang tồn tại trên một bảng hoặc một View:**

Dùng thủ tục hệ thống: `sp_helptrigger <tên bảng>`

Ví dụ 79: Xem các trigger trên bảng HANG

```
sp_helptrigger HANG
```

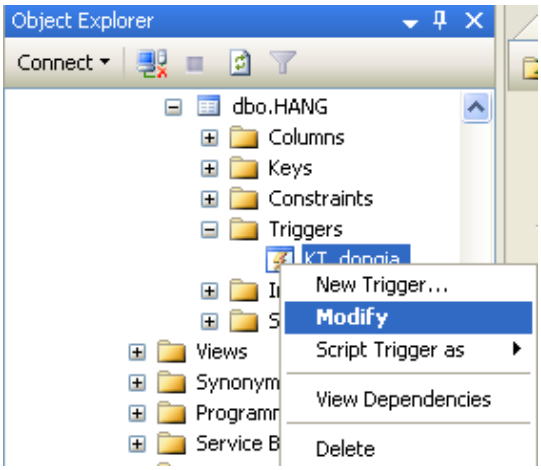
Results | Messages

	trigger_name	trigger_owner	isupdate	isdelete	isinsert	isafter	isinsteadof	trigger_schema
1	KT_dongia	dbo	1	0	0	1	0	dbo

3.7.5.1. Sửa nội dung trigger

Cách 1: Dùng công cụ SQL Server Management Studio

Từ nơi lưu trữ trigger nhấp chuột phải vào tên trigger cần xem nội dung → chọn Modify



Cách 2: Dùng lệnh theo cú pháp sau:

```
ALTER TRIGGER <tên trigger> ON <tên bảng>/<tên View>
[WITH ENCRYPTION]
FOR|AFTER|INSTEAD OF  INSERT| DELETE|UPDATE
AS
    <Các lệnh T-SQL>
```

3.7.5.2. Xoá trigger

Để xoá trigger thực hiện nhấp chuột phải vào trigger cần xoá và chọn Delete.

Ngoài ra có thể dùng lệnh: `Drop trigger <tên trigger>`

Ví dụ 80: Xoá trigger có tên là KT_dongia

```
Drop trigger KT_dongia
```

3.7.5.3. Vô hiệu hóa hoặc làm cho trigger có hiệu lực

Dùng **cú pháp** sau:

```
ALTER TABLE <tên bảng>
DISABLE|ENABLE <tên trigger>
```

3.8. Kiểu dữ liệu Cursor

3.8.1. Khái niệm

Cursor là một kiểu dữ liệu đặc biệt, hỗ trợ xử lý trên từng dòng dữ liệu cụ thể. Thông thường cursor được cài đặt kết hợp với thủ tục (stored procedure) hoặc trigger để dễ dàng sử dụng và quản lý.

Cursor chỉ dùng trong trường hợp thật sự cần thiết, là giải pháp sau cùng khi mà các giải pháp khác không thể thực hiện được.

3.8.2. Định nghĩa biến kiểu cursor

Cú pháp:

```
DECLARE  <tên cursor> CURSOR
[LOCAL | GLOBAL]
[FORWARD_ONLY | SCROLL]
[STATIC | DYNAMIC]
[READ_ONLY]
FOR
    <câu lệnh SELECT>
[FOR UPDATE [OF<danh sách cột> [,...n]]]
```

Trong đó:

- Tên cursor: Tên biến có kiểu dữ liệu cursor, tên cursor không bắt đầu bằng ký tự "@".
- LOCAL: cursor cục bộ, chỉ có thể sử dụng trong phạm vi một khối lệnh hoặc một thủ tục/hàm.
- GLOBAL: cursor toàn cục
- FORWARD_ONLY: Chỉ cho phép đọc dữ liệu trong cursor theo chiều đi tới.
- SCROLL: Cho phép đọc dữ liệu tới lui giữa các dòng bên trong cursor
- STATIC: Chỉ định việc đọc dữ liệu bên trong cursor là tĩnh. Khi đó nếu có những thay đổi bên dưới dữ liệu gốc thì những thay đổi đó sẽ không được cập nhật tự động trong dữ liệu của cursor.
- DYNAMIC: Chỉ định dữ liệu bên trong cursor là động. Khi đó việc cập nhật bên dưới dữ liệu gốc thì sẽ tự động cập nhật trong dữ liệu của cursor.

- READ_ONLY: Chỉ định dữ liệu bên trong cursor là chỉ đọc, không thể sử dụng cursor để update dữ liệu trong các bảng liên quan. Khi khai báo cursor với thuộc tính tĩnh (STATIC) thì dữ liệu trong cursor xem như là chỉ đọc.
- Câu lệnh SELECT: Dùng để chỉ đến các cột có bên trong bảng mà chúng ta cần đọc dữ liệu trong cursor
- Danh sách cột cập nhật: Chỉ định danh sách tên các cột sẽ được phép thay đổi giá trị trong cursor. Mặc định tất cả các cột trong mệnh đề SELECT sẽ được phép thay đổi giá trị nếu dữ liệu trong cursor không phải là chỉ đọc.

Mặc định, khi khai báo cursor mà không chỉ ra các tùy chọn thì cursor có các tính chất:

- Global
- Forward_Only
- Dynamic

Ví dụ 81: Định nghĩa một biến tên cursor_sv có kiểu cursor gồm mã, họ tên và ngày sinh của các sinh viên thuộc lớp có tên là '12CDTH1'

```
DECLARE cursor_sv CURSOR
DYNAMIC
FOR Select MASV,HOTEN,NGAYSINH
    From SINHVIEN,LOP
    Where SINHVIEN.MALOP=LOP.MALOP
          AND TENLOP='12CDTH1'
```

Để có thể đọc được dữ liệu bên trong cursor chúng ta cần phải mở cursor bằng lệnh OPEN như sau:

```
OPEN <tên cursor>
```

Hoạt động bên trong của lệnh này thực chất là thực hiện câu lệnh SELECT đã được chỉ định trong lệnh định nghĩa biến cursor trước đó.

Ví dụ 82: Mở cursor có tên là `cursor_sv` đã được định nghĩa trong ví dụ 81 như sau:

```
OPEN cursor_sv
```

3.8.3. Đọc và xử lý dữ liệu trong cursor

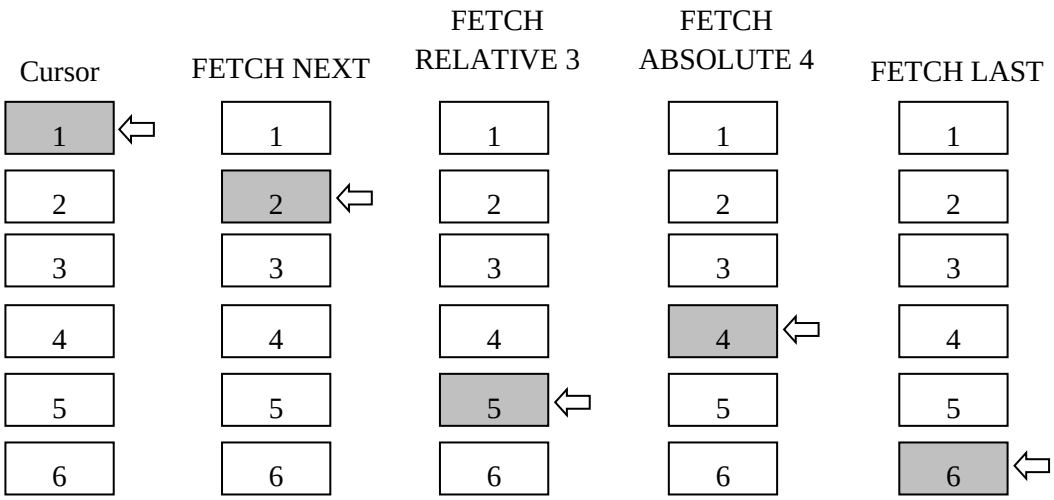
Sau khi định nghĩa và mở cursor, tiếp theo ta thực hiện đọc và xử lý dữ liệu bên trong cursor.

Cú pháp:

```
FETCH  [NEXT|PRIOR|FIRST|LAST|ABSOLUTE n|RELATIVE n]
FROM <tên cursor> [INTO <danh sách biến>]
```

Trong đó:

- NEXT: Đọc dòng dữ liệu kế tiếp dòng hiện hành
- PRIOR: Đọc dòng dữ liệu ngay trước dòng hiện hành
- FIRST: Đọc dòng dữ liệu đầu tiên
- LAST: Đọc dòng dữ liệu cuối cùng
- Danh sách biến: Danh sách tên các biến cục bộ đã được định nghĩa trước đó. Các biến này sẽ lưu trữ giá trị dữ liệu được đọc từ lệnh FETCH.



Để kiểm tra việc đọc dữ liệu thành công hay thất bại, hệ thống Microsoft SQL Server cung cấp một biến hệ thống có tên @@FETCH_STATUS có các giá trị trả về như sau:

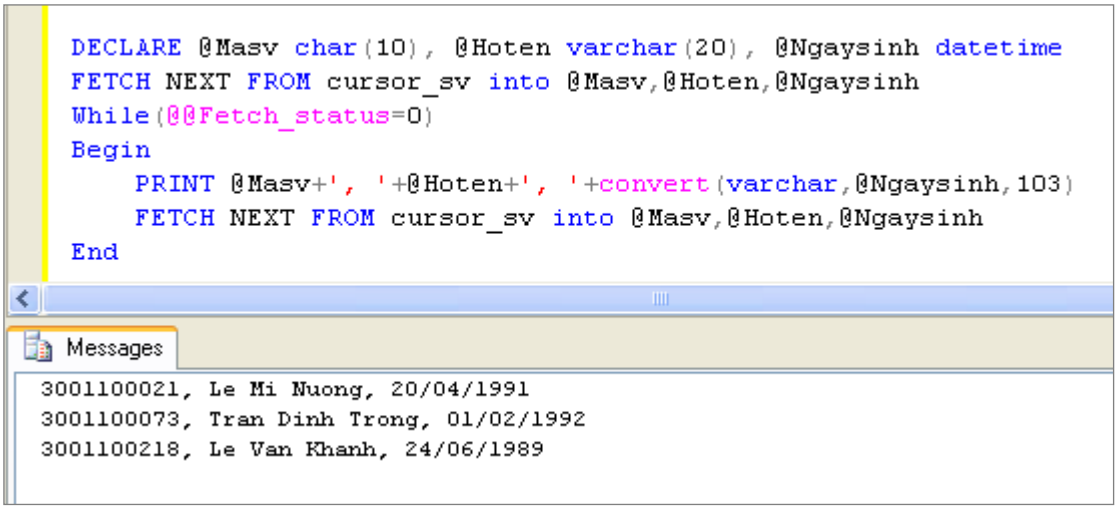
- 0: Đọc dữ liệu thành công.
- -1: Đọc dữ liệu thất bại (do cursor đang ở mẫu tin cuối cùng hay đang ở mẫu tin đầu tiên).
- -2: Đọc dữ liệu thất bại (mẫu tin không tồn tại)

✍ **Lưu ý:**

- Mặc định là Fetch Next
- Đối với cursor dạng Forward_only, chỉ có thể là Fetch Next
- Biến hệ thống @@Fetch_Status là cơ sở để biết đã duyệt đến cuối cursor hay chưa.

Ví dụ 83: Sau khi định nghĩa và mở cursor cursor_sv, để thực hiện được việc đọc và xử lý dữ liệu trong cursor này ta làm như sau:

Khai báo 3 biến @Masv, @Hoten, @Ngaysinh để lưu trữ thông tin dữ liệu tại vị trí hiện hành mà cursor trỏ đến và dùng vòng lặp While để lặp lại việc đọc dữ liệu trên cursor:



Trong cursor trên, mỗi lần đọc dữ liệu ta thực hiện xử lý bằng lệnh PRINT để in ra mã, họ tên và ngày sinh của sinh viên.

3.8.4. Đóng và hủy cursor

Sau khi đọc và xử lý xong, ta thực hiện đóng cursor hoặc huỷ (nếu không sử dụng nữa).

Đóng cursor: `CLOSE <tên cursor>`

Huỷ cursor: `DEALLOCATE <tên cursor>`

Ví dụ 84: Đóng và huỷ cursor cursor_sv:

```
CLOSE cursor_sv
DEALLOCATE cursor_sv
```

3.8.5. Tóm tắt quy trình sử dụng cursor

- Khai báo cursor
- Mở cursor
- Có thể khai báo các biến để lấy giá trị các cột của phần tử hiện hành, các biến này phải cùng kiểu dữ liệu với các cột trong câu lệnh select trong phần khai báo cursor.
- Dùng lệnh Fetch …để chuyển đến vị trí phù hợp:
  + Có thể đưa các giá trị của dòng hiện hành vào các biến thông qua

mệnh đề INTO.

+ Nếu không có mệnh đề INTO, giá trị của dòng hiện hành sẽ hiển trị ra cửa sổ kết quả sau lệnh Fetch.

– Dùng vòng lặp để duyệt cursor, kết hợp với sử dụng biến @@Fetch_Status để biết kết quả.

– Đóng cursor bằng lệnh Close. Sau khi đóng, ta vẫn có thể mở lại cursor nếu cursor chưa bị hủy.

– Hủy cursor bằng lệnh Deallocate.

Ví dụ 85: Định nghĩa biến kiểu cursor có tên là cs_sinhvien gồm 2 cột dữ liệu trong bảng sinh viên là MASV, HOTEN, mở và xử lý in các dòng dữ liệu.

```
--Định nghĩa cursor.
DECLARE cs_sinhvien CURSOR
DYNAMIC
FOR
    SELECT MASV, HOTEN
    FROM SINHVIEN
--Mở cursor
OPEN cs_sinhvien
--Xử lý dữ liệu trong cursor.
FETCH NEXT FROM cs_sinhvien
WHILE (@@FETCH_STATUS = 0)
BEGIN
  --Đọc tiếp các dòng dữ liệu nếu thành công
    FETCH NEXT FROM cs_sinhvien
END
--Đóng cursor
CLOSE cs_sinhvien
--Hủy cursor
DEALLOCATE cs_sinhvien
```

Ví dụ 86: Cho LOP(MALOP, TENLOP, SISO)

SINHVIEN(MASV, HOTEN, MALOP)

Định nghĩa cursor có tên là Tinhsiso để lấy danh sách mã lớp của bảng lớp, duyệt từng phần tử cursor để tính sĩ số của từng lớp.

```
declare Tinhsiso cursor
Dynamic
For
    Select malop FROM LOP
Open Tinhsiso
declare @malop nchar(10)
FETCH NEXT FROM Tinhsiso INTO @malop
WHILE (@@FETCH_STATUS=0)
BEGIN
    update LOP
    set sisoDK=( select count(*)
                 from sinhvien
                 where malop=@malop)
    where malop=@malop
    FETCH NEXT FROM Tinhsiso  INTO @malop
END
Close Tinhsiso
Deallocate Tinhsiso
```

3.8.6. Kết hợp cursor và thủ tục/trigger

Như đã trình bày trong phần trên, để dễ dàng sử dụng và quản lý cursor, ta cài đặt cursor lồng bên trong thủ tục/trigger. Như vậy, mỗi lần gọi thủ tục/trigger thì cursor sẽ được mở hoặc tạo mới, khi kết thúc thủ tục/trigger thì cursor sẽ được đóng lại hoặc huỷ đi.

3.8.6.1. Kết hợp cursor và thủ tục

Ví dụ 87: Cho CHITIETHD(SOHD, MAHANG, SOLUONG, DONGIA, GIAMGIA, THANHTIEN). Viết thủ tục truyền vào số hóa đơn, xuất ra tổng tiền, tiền giảm, thanh toán. Trong đó:

Giảm giá được tính như sau: khách hàng mua 3 sản phẩm thì chỉ tính tiền 2 sản phẩm.

THANHTIEN = SOLUONG * DONGIA - GIAMGIA

Tổng tiền = Sum(số lượng * đơn giá)

Tiền giảm = Sum(giảm giá)

Thanh toán = Sum(thành tiền)

Thủ tục được viết như sau:

```
CREATE PROC TINHTIEN
@SOHD CHAR(10), @TONGTIEN MONEY OUTPUT, @TIENGIAM MONEY OUTPUT, @THANHTOAN MONEY OUTPUT
AS
DECLARE  @MAHG NCHAR(10), @SOLUONG INT, @DONGIA INT
-- Khai báo cursor.
DECLARE cs_TINHTIEN CURSOR
DYNAMIC
FOR
    SELECT MAHANG,SOLUONG,DONGIA
    FROM CHITIETHD
    WHERE SOHD = @SOHD
--Mở cursor.
OPEN cs_TINHTIEN
--Xử lý dữ liệu trong cursor.
FETCH NEXT FROM cs_TINHTIEN
INTO @MAHG, @SOLUONG, @DONGIA
WHILE (@@FETCH_STATUS=0)
BEGIN
    UPDATE CHITIETHD
    SET    GIAMGIA=FLOOR(@SOLUONG/3)* @DONGIA
    WHERE SOHD=@SOHD AND MAHANG=@MAHG
    UPDATE CHITIETHD
    SET    THANHTIEN=@SOLUONG*@DONGIA-GIAMGIA
    WHERE SOHD=@SOHD AND MAHANG=@MAHG
    FETCH NEXT FROM cs_TINHTIEN
    INTO @MAHG, @SOLUONG, @DONGIA
```

```
END
--đóng cursor.
CLOSE CS_TINHTIEN
DEALLOCATE CS_TINHTIEN
SELECT @TONGTIEN=SUM(SOLUONG*DONGIA),
       @TIENGIAM=SUM(GIAMGIA),@THANHTOAN=SUM(THANHTIEN)
FROM CHITIETHD
WHERE SOHD=@SOHD
GO
--gọi thực hiện thủ tục
DECLARE  @TONGTIEN MONEY, @TIENGIAM MONEY,
         @THANHTOAN MONEY
EXEC TINHTIEN 'HD02', @TONGTIEN OUTPUT,
     @TIENGIAM OUTPUT, @THANHTOAN OUTPUT
SELECT @TONGTIEN, @TIENGIAM,@THANHTOAN
GO
```

3.8.6.2. Kết hợp cursor và trigger

Trong các ví dụ trước, ta chỉ mới xét trường hợp câu lệnh kích hoạt trigger tác động trên một dòng dữ liệu, bảng inserted và deleted chỉ chứa một dòng dữ liệu. Trong thực tế, câu lệnh insert into …select, update và delete thường có tác dụng trên nhiều dòng dữ liệu. Vậy làm thế nào để trigger hoạt động đúng trong trường hợp câu lệnh kích hoạt trigger tác động lên nhiều dòng dữ liệu?

Việc sử dụng cursor lồng vào trong trigger để duyệt và xử lý lần lượt từng dòng dữ liệu là một giải pháp khá tốt trong trường hợp này.

Ta xét ví dụ sau đây:

Ví dụ 88: Cho LOP(MALOP, TENLOP, SISO)

SINHVIEN(MASV, HOTEN, NGAYSINH, MALOP)

Giả sử ta có trigger sau:

```
CREATE TRIGGER SISO_DELETE ON SINHVIEN
FOR DELETE
AS
```

```
        UPDATE LOP
        SET SISO=SISO-1
        WHERE MALOP=(SELECT MALOP FROM DELETED)
GO
```

Trigger SISO_DELETE trên chỉ đúng trong trường hợp câu lệnh Delete xóa 1 dòng dữ liệu. Nếu ta thực hiện câu lệnh:

```
DELETE FROM SINHVIEN WHERE YEAR(NGSINH)<=1980
```

Thì lỗi sau sẽ xảy ra:

```
Msg 512, Level 16, State 1, Procedure SISO_DELETE, Line 4
Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <= , >, >= or when the subquery is used as an expression.
```

Rõ ràng lỗi trên xảy ra do câu lệnh delete tác động đến nhiều dòng dữ liệu. Để giải quyết vấn đề trên, ta sửa lại trigger như sau:

```
CREATE TRIGGER SISO_DELETE ON SINHVIEN
FOR DELETE
AS
    DECLARE CS_XOASV CURSOR
    FOR
        SELECT MASV, MALOP FROM DELETED
    OPEN CS_XOASV
    DECLARE @MASV CHAR(10), @MALOP CHAR(10)
    WHILE 0=0
    BEGIN
        FETCH NEXT FROM CS_XOASV INTO @MASV,@MALOP
        IF @@FETCH_STATUS <> 0 BREAK
        UPDATE LOP
        SET SISODK = SISODK -1
        WHERE MALOP = @MALOP
    END
    CLOSE CS_XOASV
    DEALLOCATE CS_XOASV
GO
```

Kết chương

Trong lập trình cơ sở dữ liệu cũng như các ngôn ngữ lập trình khác, việc sử dụng biến là rất quan trọng và là không thể thiếu. Biến có thể sử dụng trong một bó lệnh hay trong thủ tục, hàm, trigger, transaction. Biến do người dùng định nghĩa được khai báo bắt đầu bằng tự @, các biến hệ thống bắt đầu bằng @@. Trong SQL biến được khai báo và sử dụng một cách tường minh nghĩa là phải khai báo xong rồi mới gán giá trị. Thủ tục trong SQL có thể có tham số để nhận giá trị truyền vào gọi là tham số đầu vào và tham số nhận giá trị trả về gọi là tham số đầu ra thông qua từ khoá output, ngoài ra thủ tục còn sử dụng được lệnh return để trả về giá trị luôn là số nguyên hay biến có chứa giá trị nguyên. Hàm có hai loại thông dụng là hàm vô hướng (Scalar – Valued Function) trả về một giá trị và hàm nội tuyến (Table – Valued Function) trả về một bảng dữ liệu, loại hàm nội tuyến này rất hữu ích và có thể sử dụng uyển chuyển hơn view vì nó có thể có tham số.

Trigger là một dạng thủ tục đặc biệt có khả năng thực thi một cách tự động mỗi khi dữ liệu trên bảng liên quan với trigger được cập nhật (thêm, xoá, sửa), dùng để cài đặt các ràng buộc toàn vẹn phức tạp mà các constraint không thể thực hiện được. Ngoài ra, trigger hay thủ tục đều có thể kết hợp với cursor để giải quyết vấn đề một cách linh hoạt và uyển chuyển hơn.

Câu hỏi và bài tập chương 3

Bài tập 1:

Khai báo 2 biến như sau:

- Biến @hoten có kiểu dữ liệu là varchar(20), biến @tuoi có kiểu dữ liệu là int.
- Thực hiện gán cho biến @hoten giá trị là ‘Nguyen Van Khanh’, biến @tuoi giá trị là 20
- In giá trị của 2 biến đó ra màn hình (dùng lệnh select và lệnh print).

Bài tập 2:

Tạo database có tên là DB1, trong DB1 tạo 1 bảng có tên là SINHVIEN gồm các cột: MASV, HOTEN, NGSINH, DIEMTB. Sau đó nhập liệu vào bảng SINHVIEN.

- Thực hiện khai báo 2 biến để lưu trữ họ tên và ngày sinh của sinh viên.
- Dùng lệnh set để gán họ tên và ngày sinh của sinh viên có mã là ‘SV004’ vào 2 biến trên.
- Dùng lệnh select để gán họ tên và ngày sinh của sinh viên có mã là ‘SV004’ vào 2 biến trên.

- Dùng lệnh print để in ra giá trị của 2 biến như sau: Sinh viên Nguyen Van Khanh co ngay sinh la: 20/03/1982 (giả sử rằng sinh viên SV004 có họ tên là Nguyen Van Khanh và có ngày sinh là 20/03/1982)
- Dùng lệnh select để in ra giá trị của 2 biến trên.
- Giả sử ĐTB đã có. Hãy dùng cấu trúc if…else để thực hiện kiểm tra biến @DIEMTB như sau:

  Nếu @DIEMTB <5 xuất ra thông báo: yếu

  5≤ @DIEMTB<7 xuất ra thông báo: trung bình

  7≤ @DIEMTB<8 xuất ra thông báo: khá

  @DIEMTB≥ 8 xuất ra thông báo: gioi
- Thực hiện kiểm tra tuổi của sinh viên có mã là 'SV001', nếu tuổi của sinh viên này > 30 thì xuất ra thông tin gồm: họ tên, tuổi, điểm trung bình của sinh viên này, ngược lại xuất thông báo: *Sinh vien nay duoi 30 tuoi*
- Dùng cấu trúc if exists …else để kiểm tra nếu có sinh viên có điểm trung bình >5 thì xuất ra tổng số các sinh viên đó, ngược lại xuất ra thông báo: *khong co sinh vien tren trung binh*

Bài tập 3: Cho CSDL gồm các bảng như sau:

LOP(MALOP, TENLOP, SISO)

SINHVIEN(MASV, HOTEN, NGSINH, GIOITINH, QUEQUAN, MALOP, DIEMTB, XEPLOAI)

MONHOC(MAMH, TENMH, SOTC, BATBUOC)

KETQUA(MASV, MAMH, HOCKY, DIEMTHI)

Yêu cầu:

a/ Tạo database QLSV sau đó tạo các bảng dữ liệu như trên.

b/ Tạo các ràng buộc khóa ngoại(nếu có) và lược đồ Diagram.

c/ Viết thủ tục hiện cập nhật SISO trên bảng LOP dựa vào bảng SINHVIEN.

d/ Viết thủ tục thực hiện cộng 1 điểm cho sinh viên khi truyền vào 3 tham số là mã sinh viên và tên môn học và học kỳ.

e/ Viết thủ tục in ra họ tên và tên lớp của sinh viên khi truyền vào tham số MASV.

f/ Viết thủ tục trả về họ tên và tổng số môn học mà sinh viên đó học khi truyền vào 2 tham số: mã sinh viên và học kỳ

g/ Viết thủ tục khi truyền vào tên môn học và học kỳ sẽ trả về mã môn học, số tín chỉ và tổng số sinh viên học môn học trong học kỳ đó

h/ Viết thủ tục khi truyền vào 3 tham số: mã sinh viên, tên môn học và học kỳ sẽ trả về 'chua dang ky' nếu như sinh viên đó chưa đăng ký môn học, trả về 'dat' nếu điểm môn đó >=5, trả về 'khong dat' neu điểm của môn đó <5.

i/ Viết thủ tục trả về điểm trung bình của sinh viên khi truyền vào tham số mã sinh viên

(HD: Điểm trung bình =(số tín chỉ1 * điểm môn học1+số tín chỉ2*điểm môn hoc2+…+số tín chỉ n * điểm môn học n)/tổng số tín chỉ)

j/ Viết hàm cho các câu c,d,e,f,g,h,i

k/ Viết hàm nội tuyến trả về table gồm: mã sinh viên, họ tên, tên lớp, tổng số môn học. Với tham số truyền vào là 'lop A' Thực hiện truy vấn trên hàm (liệt kê mã họ tên và tổng số môn học của các sinh viên)

Bài tập 4: Cho CSDL gồm các bảng như sau:

DOCGIA(<u>MADG</u>, TENDG, DIACHI)

SACH(<u>MASH</u>, TENSH, LOAI, NXB, NAMXB, TACGIA, TINHTRANG)

MUONSACH(<u>MADG, MASH, NGAYMUON</u>, NGAYTRA)

Yêu cầu:

a/ Tạo một database mới sau đó tạo các bảng dữ liệu như trên.

b/ Tạo các ràng buộc khóa ngoại(nếu có) và lược đồ Diagram.

c/ Viết thủ tục truyền vào tham số mã đọc giả sẽ trả về tên đọc giả và địa chỉ.

d/ Viết thủ tục truyền vào tham số mã sách sẽ trả về tên sách, năm xuất bản và tác giả.

e/ Viết thủ tục truyền vào tham số mã đọc giả sẽ trả về số lượng sách mà đọc giả đang mượn, nếu không có sách nào đang mượn thì trả về 0 (ghi chú: sách chưa trả thì ngày trả có giá trị NULL)

f/ Viết thủ tục truyền vào mã sách sẽ trả về tên đọc giả nếu đọc giả đang mượn sách đó, nếu sách hiện không có đọc giả nào mượn thì trả về 'chua muon'

g/ Viết thủ tục truyền vào tham số mã đọc giả và ngày/tháng/năm sẽ trả số sách mà đọc giả mượn trong ngày/tháng/năm đó, nếu đọc giả không có mượn sách trong ngày đó thì trả về 0.

h/ Viết thủ tục truyền vào mã sách sẽ trả về ngày/tháng/năm mà sách đó được mượn gần nhất.

i/ Viết lại các câu: e, f, g, h bằng cấu trúc hàm.

Bài tập 5: Cho CSDL gồm các bảng như sau:

KHACH(MAKH, TENKH, DCHI, DTHOAI)

NHASX(MANSX, TENNSX, DCHI, DTHOAI)

NHACC(MANCC, TENNCC, DCHI, DTHOAI)

PHIEUNHAP(MAPN, NGAYNHAP, MANCC, TIENNHAP)

HANG(MAHG, TENHG, DVT, SOLUONG, MANSX, MANCC, TINHTRANG)

CHITIETPN(MAPN, MAHG, SOLUONG, GIANHAP, THANHTIEN)

HOADON(MAHD, NGAYBAN, TENNV, MAKH, TIENBAN, GIAMGIA, THANHTOAN)

CHITIETHD(MAHD, MAHG, SOLUONG, GIABAN, THANHTIEN)

DONGIA(MAHG, NGAYCN, GIA)

Yêu cầu:

a/ Tạo một database mới sau đó tạo các bảng dữ liệu như trên.

b/ Tạo các ràng buộc khóa ngoại(nếu có) và lược đồ Diagram.

c/ Viết thủ tục thực hiện việc cập nhật THANHTIEN trên bảng CHITIETHD và bảng CHITIETPN.

d/ Viết thủ tục thực hiện cập nhật TIENNHAP và TIENBAN trên bảng PHIEUNHAP và bảng HOADON.

e/ Viết thủ tục truyền vào tham số mã khách hàng sẽ in ra danh sách các hóa đơn (mã hóa đơn, tổng trị giá) của khách hàng đó

f/ Viết thủ tục truyền vào tham số mã hóa đơn sẽ trả về ngày lập và trị giá của hóa đơn đó.

g/ Viết thủ tục truyền vào tham số mã hàng sẽ trả về tên hàng, tên nhà sản xuất và tên nhà cung cấp tương ứng.

h/ Để kiểm tra một khách hàng thuộc loại nào (‘VIP’, ‘KH thành viên’, ‘KH thân thiết’) cần viết một thủ tục truyền vào tham số mã khách hàng sẽ trả về ‘VIP’ nếu doanh số ≥10.000.000; ‘KH thành viên’ nếu 6.000.000 ≤doanh số<10.000.000; ‘KH thân thiết’ nếu doanh số <6.000.000 (ghi chú: Doanh số là số tiền mà khách mua hàng).

i/ Viết thủ tục truyền vào mã hàng sẽ trả về ngày cập nhật đơn giá gần nhất.

j/ Viết lại các câu g, f bằng cấu trúc hàm

Bài tập 6: Cho CSDL gồm các bảng như sau:

LOP(MALOP, TENLOP, SISO)

SINHVIEN(MASV, HOTEN, NGSINH, GIOITINH, QUEQUAN, MALOP, DIEMTB, XEPLOAI)

MONHOC(MAMH, TENMH, SOTC, BATBUOC)

KETQUA(MASV, MAMH, HOCKY, DIEMTHI)

Yêu cầu:

a/ Viết trigger thực hiện cập nhật sĩ số SISO trên bảng LOP mỗi khi thêm, xóa hay sửa dữ liệu trên bảng SINHVIEN

b/ Viết trigger thực hiện tính điểm trung bình DIEMTB trên bảng SINHVIEN mỗi khi thêm, xóa hay sửa dữ liệu trên bảng KETQUA

c/ Dùng cấu trúc trigger viết ràng buộc toàn vẹn thực hiện kiểm tra mỗi sinh viên chỉ được đăng ký tối đa 3 môn trong mỗi học kỳ.

d/ Dùng cấu trúc trigger viết ràng buộc toàn vẹn thực hiện kiểm tra mỗi sinh viên chỉ đăng ký tối đa 10 tín chỉ của môn học bắt buộc trong mỗi học kỳ.

e/ Viết trigger thực hiện điền giá trị vào cột XEPLOAI dựa vào cột DIEMTB với điều kiện như sau:

DIEMTB<5 :'yeu'

5≤DIEMTB<7 :'trung binh'

7≤DIEMTB<8:'kha'

DIEMTB≥8 :'Giỏi'

Bài tập 7: Cho CSDL gồm các bảng như sau:

SACH(MASH, TENSH, TACGIA, LOAI, TINHTRANG)

DOCGIA(MADG, TENDG, TUOI, PHAI, DIACHI)

MUONSACH(MADG, MASH, NGAYMUON, NGAYTRA)

Yêu cầu:

a/ Viết trigger kiểm tra tuổi của độc giả phải >=15

b/ Viết trigger kiểm tra phái của độc giả phải là 'Nam' hay 'Nu'

c/ Viết trigger kiểm tra loại sách phải thuộc trong các loại như: Khoa hoc tu nhien, Xa hoi, Kinh te, Truyen

d/ Dùng lệnh xoá các ràng trigger trên

e/ Viết trigger kiểm tra khi thêm hay sửa dữ liệu trên bảng MUONSACH thì ngày trả >= ngày mượn

f/ Viết trigger kiểm tra cập nhật TINHTRANG trên bảng SACH là 'Da muon' mỗi khi thêm sách vào bảng MUONSACH (tức là hành động cho mượn sách). Khi thêm dòng dữ liệu vào bảng MUONSACH thì NGAYTRA để trống.

g/ Viết trigger kiểm tra cập nhật TINHTRANG trên bảng SACH là 'Chua muon' mỗi khi sách đó được trả (tức là khi cập nhật NGAYTRA vào bảng MUONSACH)

h/ Viết trigger kiểm tra kiểm tra nếu số sách chưa trả >=3 thì không được mượn tiếp. (HD: sách chưa trả nghĩa là NGAYMUON có giá trị NULL, hành động cho mượn sách có nghĩa là cho thêm dòng dữ liệu vào bảng MUONSACH)

Bài tập 8: Cho CSDL gồm các bảng như sau:

KHACHHG(MAKH, TENKH, DCHI, DTHOAI)

NHASX(MANSX, TENNSX, DCHI, DTHOAI)

NHACC(MANCC, TENNCC, DCHI, DTHOAI)

PHIEUNHAP(MAPN, NGAYNHAP, MANCC, TIENNHAP)

HANG(MAHG, TENHG, DVT, SOLUONG, MANSX, TINHTRANG)

CHITIETPN(MAPN, MAHG, SOLUONG, GIANHAP, THANHTIEN)

HOADON(MAHD, NGAYBAN, TENNV, MAKH, TIENBAN, GIAMGIA, THANHTOAN)

CHITIETHD(MAHD, MAHG, SOLUONG, GIABAN, THANHTIEN)

DONGIA(MAHG, NGAYCN, GIA)

Yêu cầu:

a/ Viết trigger thực hiện tự động cập nhật lại THANHTIEN trên bảng CHITIETHD và bảng CHITIETPN đồng thời cập nhật TIENNHAP và TIENBAN trên bảng PHIEUNHAP và HOADON mỗi khi thêm dữ liệu vào bảng CHITIETPN hay bảng CHITIETHD

b/ Viết trigger thực hiện việc cập nhật TINHTRANG trong bảng HANG thành 'Da ban' mỗi khi bán mặt hàng đó ra (thêm dữ liệu vào bảng CHITIETHD). Nếu SOLUONG của mặt hàng đó trên bảng HANG bằng 0 thì TINHTRANG chuyển thành 'Het hang'. Nếu số lượng bán ra lớn hơn số lượng hiện có (trên bảng HANG) thì xuất ra câu thông báo'Khong du hang ban'.

c/ Viết trigger thực hiện việc tăng SOLUONG trên bảng HANG mỗi khi nhập mặt hàng đó vào(nhập dữ liệu vào bảng CHITIETPN).

d/ Viết trigger thực hiện việc giảm SOLUONG trên bảng HANG mỗi khi bán mặt hàng đó ra(nhập dữ liệu vào bảng CHITIETHD).

e/ Viết trigger thực hiện lấy ngày hiện tại cho cột NGAYNHAP mỗi khi thêm dữ liệu vào bảng PHIEUNHAP.

f/ Viết trigger thực hiện lấy ngày hiện tại cho cột NGAYBAN mỗi khi thêm dữ liệu vào bảng HOADON.

g/ Viết trigger thực hiện việc lấy đơn giá mới nhất cho cột GIABAN trên bảng CHITIETHD mỗi khi thêm dữ liệu vào bảng này (dựa vào MAHG, NGAYCN và GIA trên bảng DONGIA)

h/ Viết trigger thực hiện điền vào cột giảm giá trên bảng HOADON mỗi khi bán hàng là '5%' nếu 200000 ≤ TIENBAN< 500000. Nếu TIENBAN ≥ 500000 thì điền vào '10%'. Nếu TIENBAN <200000 thì điền vào '0'. Đồng thời tính tiền cho cột THANHTOAN = TIENBAN- TIENBAN* (tỉ lệ giảm giá)

Bài tập 9: Cho CSDL gồm các bảng như sau:

NHACUNGCAP(MANCC, TENNCC, DCHI, DTHOAI)

MATHANG(MAMH, TENMH, DVT, QUYCACH, SLTON, DG)

CUNGUNG(MANCC, MAMH)

DATHANG(SODH, NGAYDH, MANCC, SL_MATHANG, GHICHU, THANHTIEN)

CTDH(SODH, MAMH, SLD, DGD)

GIAOHANG(SOGH, NGAYGH, SODH)

CTGH(SOGH, MAMH, SLG, DGG)

Yêu cầu:

1. Cài đặt các ràng buộc toàn vẹn sau:
   a. Số lượng tồn của một mặt hàng luôn luôn > 0
   b. Đơn vị tính thuộc một trong các đơn vị sau: lốc, chai, thùng, túi, bao, bình, hộp, hũ, gói, kg
   c. Quy cách đóng gói thuộc một trong các quy cách sau: chai, gói, hộp, thùng
   d. Trong một lần đặt hàng, nhà cung cấp có thể giao hàng tối đa 3 lần
   e. Không được phép giao hàng trễ hơn 1 tuần so với ngày đặt hàng
   f. Chỉ có thể đặt các mặt hàng mà nhà cung cấp đó cung ứng
   g. Chỉ được giao các mặt hàng mà khách hàng có đặt

h. Tổng số lượng giao của một mặt hàng trong một lần đặt hàng phải nhỏ hơn hoặc bằng số lượng đặt

i. Số mặt hàng trong đặt hàng phải bằng số các mặt hàng được đặt trong chi tiết đặt hàng của cùng một lần đặt hàng

j. Cập nhật lại số lượng tồn của mặt hàng một cách tự động mỗi khi nhà cung cấp giao hàng

k. Tự động cập nhật thành tiền của một lần đặt hàng mỗi khi thêm 1 mặt hàng vào đơn đặt hàng đó.

2. Cài đặt các stored procedure sau:

a. Danh sách các đơn đặt hàng của một nhà cung cấp, với MANCC là tham số input

b. Tính thành tiền của một lần giao hàng, với SOGH là tham số input và THANHTIEN là tham số output

c. Dùng từ khóa Return trả về số lượng mặt hàng mà một nhà cung cấp có thể cung ứng

d. Viết hàm trả về 1 table gồm các thông tin sau: MAMH, SLD, DGD của một đơn đặt hàng bất kỳ

Bài tập 10:

Yêu cầu:

a/ Viết lệnh tạo và nhập dữ liệu vào bảng THISINH sau:

SBD	HOTEN	KHUVUC	DIEMTHEM
A1001	Tran Thanh Nam	1	
B1002	Nguyen Ngoc Phu	2	
C1001	Vo Van Viet	1	
A1002	Trinh Dinh Dong	3	

b/ Dùng cấu trúc cursor để cập nhật giá trị vào cột DIEMTHEM với điều kiện như sau:

KHUVUC = 1: DIEMTHEM = 0

KHUVUC = 2: DIEMTHEM = 0.5

KHUVUC = 3: DIEMTHEM = 1

Bài tập 11: Yêu cầu:

Tạo và nhập bảng dữ liệu như sau:

MASV	HOTEN	DIEM_KT	DIEM_GK	DIEM_CK	DIEM_TK
3001090113	Lam Bich Van	8	7	8	
3001090344	Nguyen Thanh Nam	3	5	7	
3001100021	Le Mi Nuong	8	2	9	
3001100022	Tran Dinh Trong	6	4	5	
3001100023	Le Van Khanh	7	3	8	
3001090345	Chu Bao Chau	6	5	9	

Hãy viết lệnh định nghĩa cursor có tên là TinhDiem gồm các thuộc tính: mã sinh viên, điểm kiểm tra, điểm giữa kỳ, điểm cuối kỳ. Thực hiện xử lý trên cursor như sau: Mỗi khi cursor di chuyển đến mẫu tin kế tiếp thì tính điểm tổng kết môn (DIEM_TK) của sinh viên tương ứng theo công thức:

DIEM_TK = DIEM_KT*20%+DIEM_GK*30%+ DIEM_CK*50%.

Chương 4
BẢO MẬT VÀ AN TOÀN DỮ LIỆU

Mục tiêu:

- *Hiểu được một số tác vụ cần thiết cho việc quản trị cơ sở dữ liệu của Microsoft SQL Server.*
- *Thực hành sao lưu và phục hồi cơ sở dữ liệu.*
- *Trình bày và thực hiện được việc cấp quyền và bảo mật hệ thống.*

4.1. Sao lưu và phục hồi cơ dữ liệu

Một cơ sở dữ liệu khi đã được thiết kế, xây dựng và đưa vào sử dụng đôi khi gặp một số sự cố ngoài ý muốn như:

- Đĩa chứa Data File hay Transaction Log File bị mất, bị hư hỏng.
- Server bị hỏng.
- Những thảm họa tự nhiên như bão lụt, động đất, hỏa hoạn.
- Các thiết bị dùng để Backup - Restore bị đánh cắp hay hư hỏng.
- Những lỗi do vô ý của user như lỡ tay xóa, thao tác sai làm hư cơ sở dữ liệu.
- Những hành vi mang tính phá hoại như cố ý đưa vào những thông tin sai lạc.
- Bị hack (nếu server có kết nối với internet).

Những sự cố này có thể làm ảnh hưởng đến cơ sở dữ liệu và sẽ gây ra những thiệt hại nhất định. Để tránh và hạn chế tối đa mất mát dữ liệu do các sự cố nêu trên, là một người quản trị hệ thống cơ sở dữ liệu, chúng ta cần phải biết cách sao lưu (backup) và khôi phục (restore) dữ liệu, sắp xếp lịch trình sao lưu dữ liệu một cách hợp lý để bảo quản cơ sở dữ liệu của mình một cách an toàn nhất.

4.1.1. Các mô hình phục hồi

4.1.1.1. Full Recovery Model

Đây là mô hình cho phép phục hồi dữ liệu với ít rủi ro nhất. Nếu một CSDL được thiết lập ở mô hình phục hồi này thì tất cả các hoạt động của nó đều được ghi vào Transaction Log File. Khi có sự cố xảy ra thì ta có thể phục hồi lại dữ liệu ngược trở lại tới một thời điểm trong quá khứ. Khi Data File bị hư, nếu ta có thể sao lưu được Transaction Log File thì ta có thể phục hồi CSDL đến thời điểm transaction gần nhất được commited.

Khuyết điểm của mô hình phục hồi đầy đủ là Transaction Log File có thể rất lớn.

4.1.1.2. Bulk Logged Recovery Model

Đây là mô hình phụ trợ cho mô hình phục hồi đầy đủ, khi mô hình phục hồi đầy đủ được sử dụng đôi khi làm giảm hiệu suất hệ thống vì dung lượng tập tin log có thể tăng lên quá lớn. Trong những tình huống này, cơ sở dữ liệu có thể được cấu hình để tối thiểu hoạt động của tập tin log lớn bằng cách thay đổi mô hình phục hồi Bulk Logged Recovery Model. Trong mô hình này, tối thiểu hóa tập tin log gồm những hoạt động như sau:

- Index creation
- Index rebuild
- Bulk copy operations
- BULK INSERT
- SELECT INTO
- LOB operations

Tối thiểu hóa tập tin log có nghĩa là những hoạt động trong danh sách liệt kê ở trên sẽ được ghi lại nhưng các dòng cụ thể thì không được ghi. Ngoài mẫu tin trong tập tin log dùng để ghi lại những sự thay đổi, một mẫu tin vật lý mở rộng được cấp phát hoặc bị ảnh hưởng bởi hoạt động được ghi vào transaction log, đến khi sự kiện sao lưu log diễn ra thì những ảnh hưởng vật lý sẽ được sao chép vào backup log.

Mô hình này giữ cho tập tin log nhỏ gọn hơn nhưng tập tin log backup có thể sẽ rất lớn bởi vì tập tin log backup dựa trên dữ liệu vật lý. Nếu ổ đĩa bị hỏng thì việc backup log sẽ không thành công.

Tóm lại, ở mô hình này các hoạt động mang tính hàng loạt như Bulk Insert, bcp, Create Index, WriteText, UpdateText chỉ được ghi tối thiểu vào Transaction Log File đủ để cho biết là các hoạt động này có diễn ra mà không ghi toàn bộ chi tiết như trong mô hình Full Recovery Mode. Các hoạt động khác như Insert, Update, Delete vẫn được log đầy đủ để dùng cho việc phục hồi sau này.

4.1.1.3. Simple recovery model

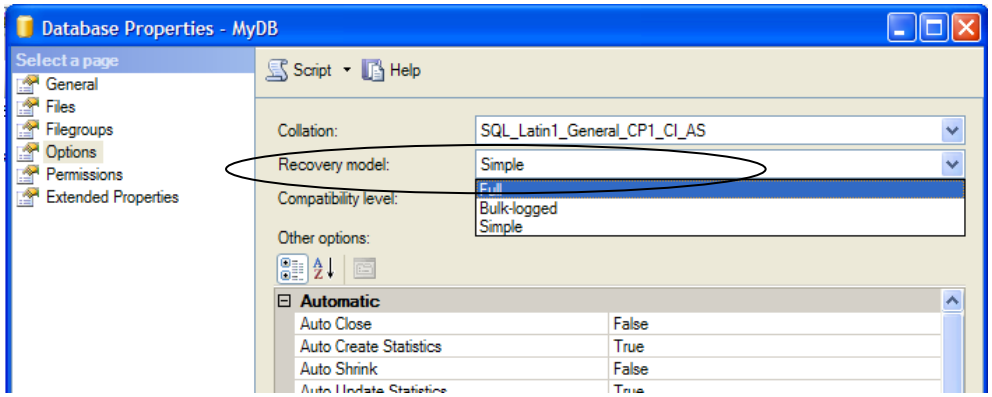
Trong mô hình này, phần không hoạt động của log được cắt gọn mỗi khi SQL Server phát ra lệnh checkpoint. Lệnh này được SQL Server thi hành định kỳ. Phần không hoạt động của log chủ yếu là phần transaction cũ.

Ưu điểm của mô hình này là giảm chi phí quản trị transaction log bởi vì phần không hoạt động của log bị xóa mỗi lần thực thi lệnh checkpoint do đó dung lượng của tập tin log không tăng nên không cần phải quản lý. Tuy nhiên với cách hoạt động này thì không thể dùng tập tin transaction log cho việc phục hồi dữ liệu bởi vì nó không có đầy đủ tất cả các record của tất cả các transaction ghi lại sự thay đổi trên database.

Nói cách khác, mô hình này chỉ cho phép phục hồi tới thời điểm sao lưu gần nhất mà không thể phục hồi tới một thời điểm trong quá khứ.

Để chọn mô hình phục hồi trong SQL Server 2008 thực hiện như sau:

Click chuột phải vào cơ sở dữ liệu → chọn Properties → chọn Options → Chọn mô hình phục hồi thích hợp.



Hình 4.1. Hộp thoại Database Properties – Thiết lập mô hình phục hồi

Hoặc cũng có thể dùng lệnh để thiết lập mô hình phục hồi theo cú pháp sau:

```
Alter database <tên database>
Set recovery <simple/full/bulk_logged>
```

Ví dụ 1.

```
Alter database QLSV
Set recovery full
```

4.1.2. Sao lưu cơ sở dữ liệu (Backup Database)

Việc sao lưu CSDL có thể thực hiện trong lúc CSDL đang hoạt động, không cần phải ngắt kết nối hay tắt bất cứ dịch vụ nào, sao lưu có thể thực hiện lưu dữ liệu lên đĩa hoặc băng từ, để thực hiện lưu trữ trên băng từ thì thiết bị băng từ phải được gắn vào Database Server.

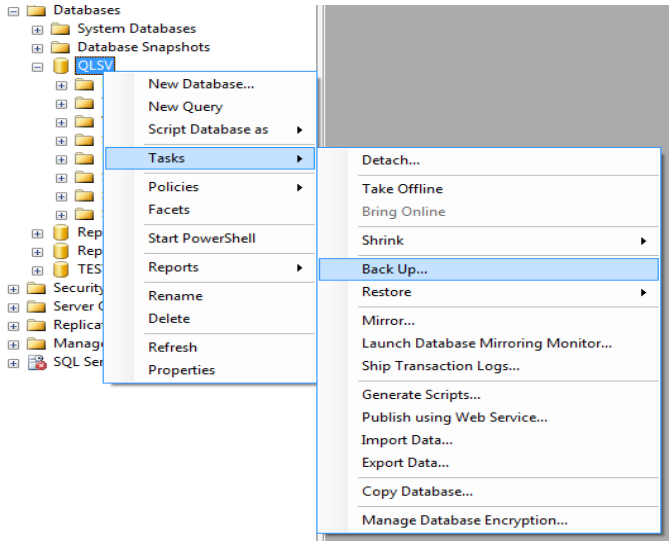
Có các hình thức sao lưu như sau:

4.1.2.1. Full Backup

Là loại sao lưu thực hiện dễ dàng nhất. Hình thức này sẽ sao lưu tất cả dữ liệu trong CSDL, và có thể sử dụng cho mọi mô hình phục hồi. Ưu điểm của loại hình này là tính đơn giản, tuy nhiên sẽ mất nhiều thời gian hơn các phương pháp khác trong trường hợp CSDL lớn.

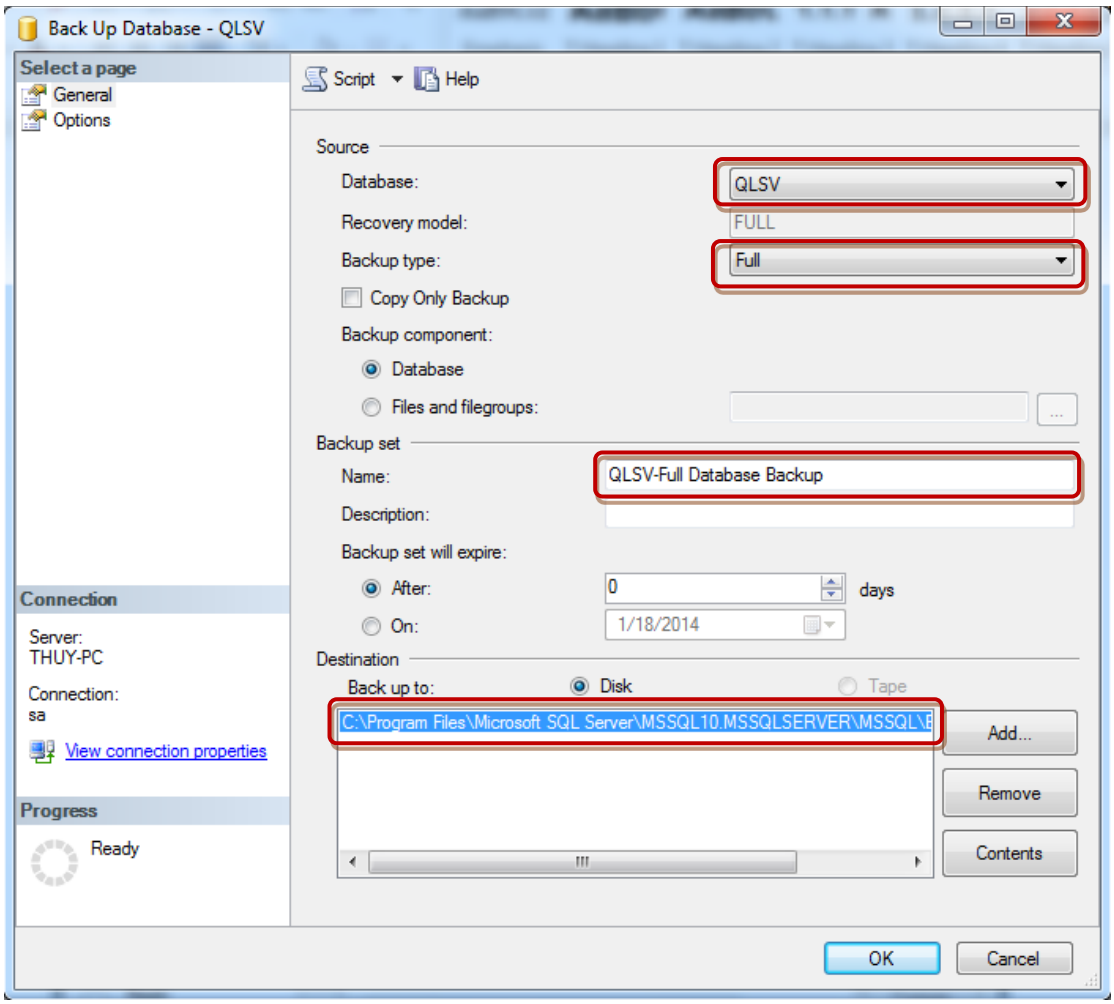
Thực hiện sao lưu cơ sở dữ liệu QLSV bằng hình thức Full Backup ta làm như sau:

Click chuột phải vào cơ sở dữ liệu cần sao lưu → chọn Tasks → chọn Back up như hình dưới đây:



Hình 4.2. Thiết lập lệnh sao lưu dữ liệu

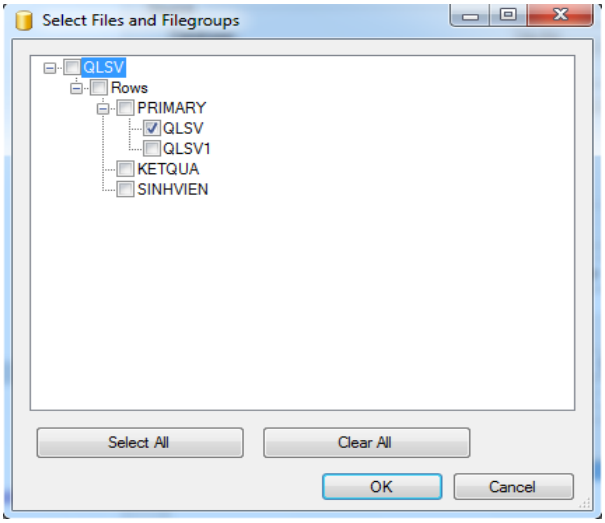
Trên cửa sổ Back up Database, chọn các thuộc tính phù hợp để tiến hành sao lưu:



Hình 4.3. Hộp thoại thiết lập thông số khi sao lưu

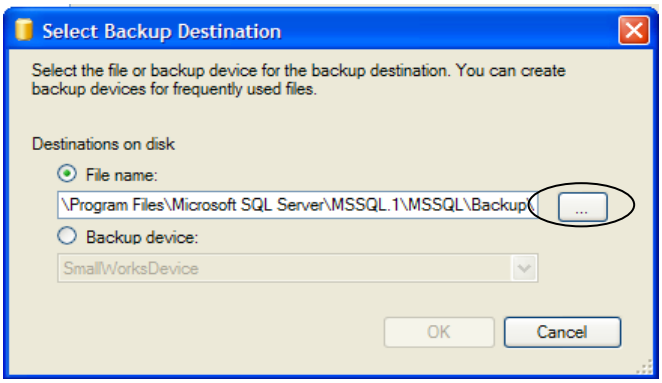
Trong đó:

- **Database:** Cơ sở dữ liệu cần backup.
- **Backup type:** chọn Full
- **Name:** Tên của bản backup này, có thể đặt tên tùy ý.
- **Backup component***:* Trong mục này chọn một trong hai lựa chọn sau:
  + ***Database*:** Backup tất cả các file hay nhóm file đã được tạo của database này.
  + ***Files and Filegroups****:* Nếu cơ sở dữ liệu rất lớn và ta chỉ muốn backup một hay một số file chứ không backup tất cả các file thì chọn mục này và chỉ định file cần backup như hình bên dưới. Trong hộp thoại này ta cũng có thể chọn chức năng backup tất cả các file.



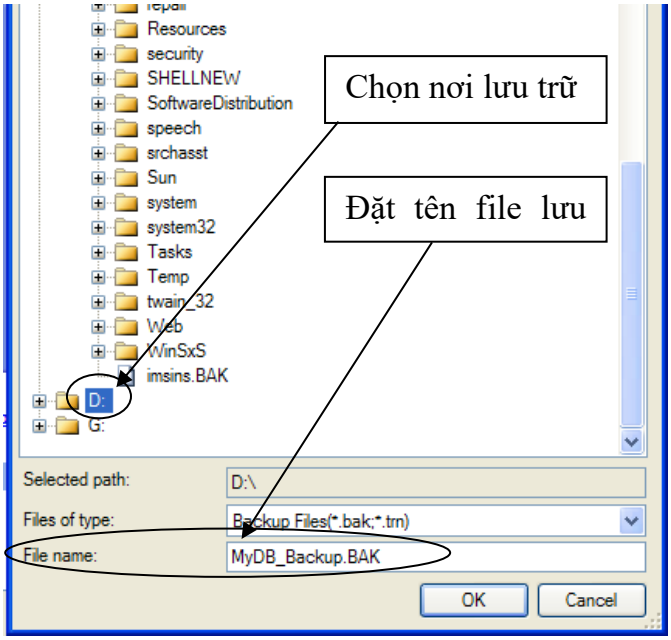
Hình 4.4. Hộp thoại Select Files and Filegroups

- **Backup set will expire:** Có thể được sử dụng khi backup vào băng hoặc tạo các nhóm backup "Media Sets". Trong ví dụ này, ta sẽ backup vào một ổ đĩa cứng vì vậy mặc định sẽ là "sau 0" ngày.
- **Destination:** Chọn nơi lưu tập tin sao lưu. Sao lưu vào đĩa hoặc sao lưu vào băng. Nếu muốn chọn vị trí lưu mới, nhấn nút Add để chọn đường dẫn, hộp thoại Select Backup Destination xuất hiện → chọn File name → nhấn nút  để chọn đường dẫn lưu file.



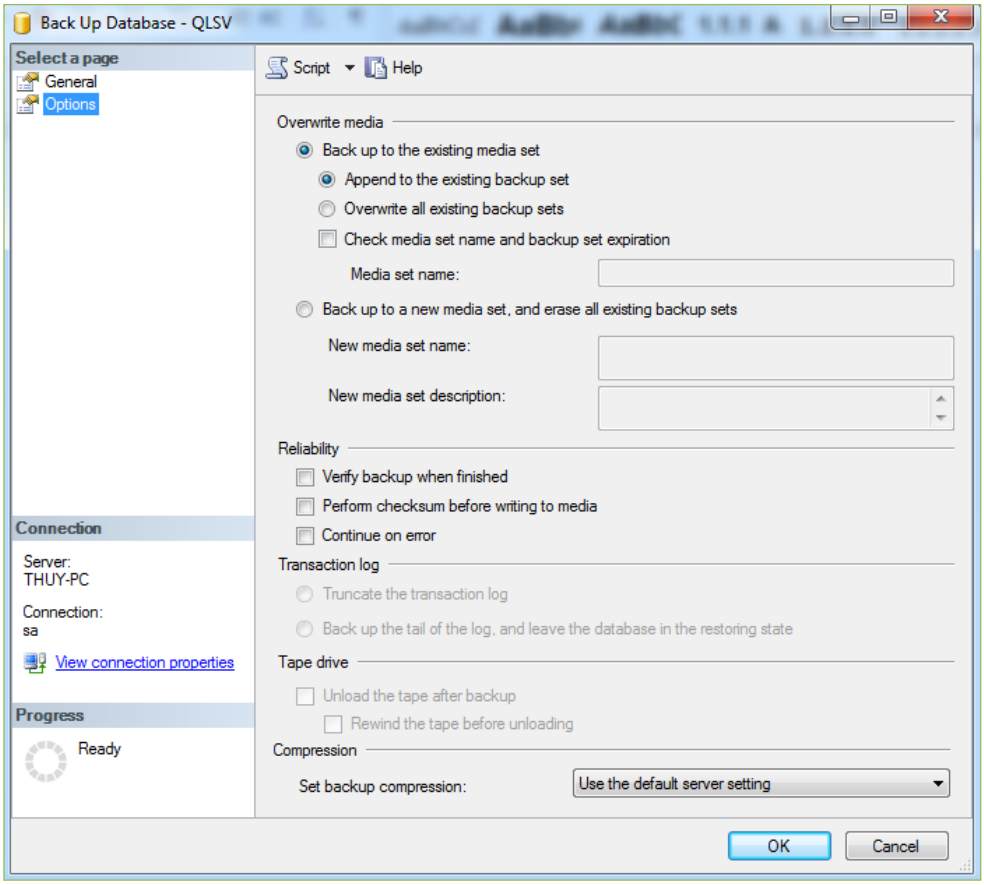
Hình 4.5. Hộp thoại chọn vị trí lưu file sao lưu

Chọn nơi lưu trữ và đặt tên cho file sao lưu



Hình 4.6. Hộp thoại chọn vị trí và đặt tên file sao lưu

Để kiểm tra các tùy chọn, chọn **Options** từ phía trên bên trái như hình sau:



Hình 4.7. Hộp thoại thiết lập các tùy chọn khi sao lưu

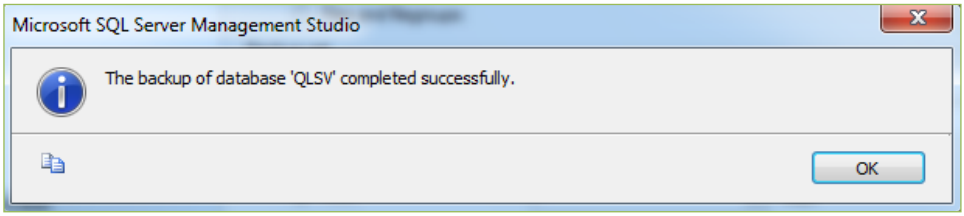
– Mục **Overwrite media:**

+ *Append to the existing backup set:* thêm một file backup mới

+ *Overwrite All existing backup sets*: ghi đè lên file đang tồn tại.

Sau khi chọn xong các thuộc tính, hệ thống sẽ tiến hành sao lưu CSDL đã chọn.

Khi quá trình sao lưu hoàn tất, hộp thoại thông báo sẽ xuất hiện và công việc sao lưu đã thành công.



Hình 4.8. Hộp thoại thông báo sao lưu thành công

Ngoài cách sao lưu trên, có thể thực hiện sao lưu bằng lệnh như sau:

```
BACKUP DATABASE QLSV
TO DISK='D:\QLSV_Full.bak'
WITH INIT, DESCRIPTION ='Backup database QLSV'
```

Trong đó : lệnh BACKUP khi có lựa chọn "WITH INIT" thì mỗi lần thực hiện sẽ ghi đè lên file hiện tại.

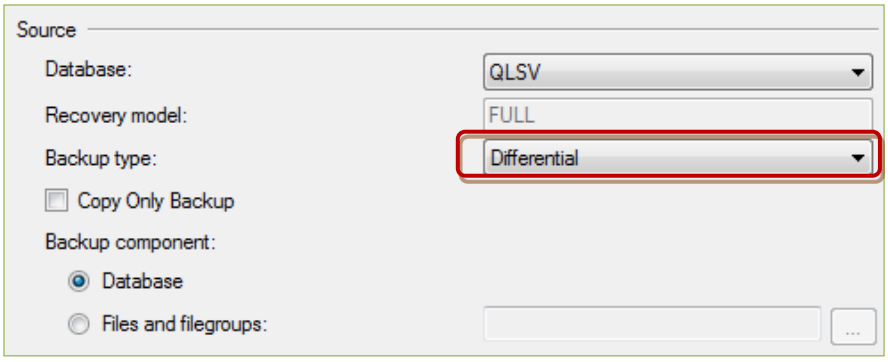
Sau khi cho thực hiện lệnh Backup Database như trên, ta sẽ thấy mấy dòng thông báo cho biết quá trình backup đã thành công như dưới đây:

```
Processed 184 pages for database 'QLSV', file 'QLSV' on file 1.

Processed 8 pages for database 'QLSV', file 'QLSV1' on file 1.

Processed 1 pages for database 'QLSV', file 'QLSV_log' on file 1.

BACKUP DATABASE successfully processed 193 pages in 0.240 seconds (6.282 MB/sec).
```

4.1.2.2. Differential Backup

Được sử dụng để sao lưu chỉ những phần dữ liệu bị thay đổi kể từ lần sao lưu Full Backup cuối cùng, do đó thời gian sao lưu sẽ ngắn hơn Full Backup và dung lượng file sao lưu cũng giảm đáng kể. Tuy nhiên nếu sao lưu dạng này nhiều lần liên tiếp sẽ dần dần trở nên lớn hơn. Nếu chỉ 1 byte dữ liệu bị thay đổi trong extent (64KB), Differential Backup sẽ sao lưu toàn bộ extent. Differential Backup được sử dụng cho bất kỳ mô hình phục hồi nào với điều kiện phải có một bản sao lưu Full Backup trước đó.

Thực hiện sao lưu cơ sở dữ liệu QLSV bằng hình thức Differential Backup ta làm tương tự như cách làm của Full Backup, chỉ khác ở chỗ tại mục Backup type cần lựa chọn loại Differential.



Hình 4.9. Hộp thoại chọn hình thức sao lưu Differential

Ngoài cách sao lưu trên, có thể thực hiện bằng lệnh như sau:

```
BACKUP DATABASE QLSV
TO DISK='D:\QLSV_Diff.bak'
WITH DIFFERENTIAL;
```

4.1.2.3. Transaction log backup

Tập tin Transaction Log có chức năng ghi lại tất cả những thay đổi trong cơ sở dữ liệu nên để giảm khả năng mất dữ liệu xuống mức thấp nhất thì tập tin này cần phải được sao lưu lại định kỳ để có thể phục hồi các thao tác nếu cơ sở dữ liệu xảy ra sự cố. Để khôi phục thành công cơ sở dữ liệu, thường phải sử dụng nhiều file Transaction Log Backup đã được tạo và chúng phải được khôi phục theo trật tự khi tạo. Nếu một file Transaction Log Backup nào đó bị lỗi, chúng ta sẽ không thể thực hiện khôi phục bất kì file Transaction Log Backup nào sau file lỗi đó. Chúng cần được khôi phục theo thứ tự và không thể có sự gián đoạn nào.

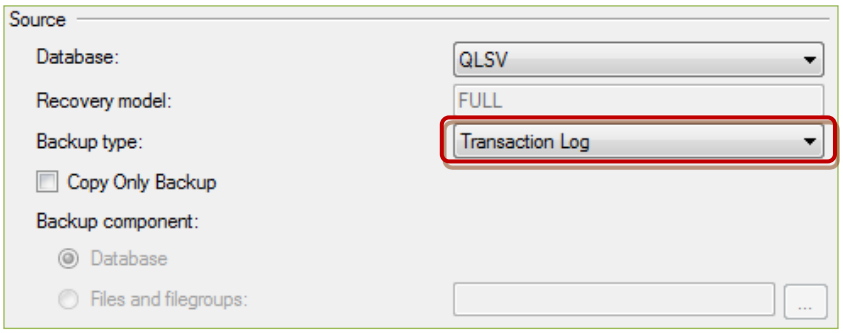
Khi tạo transaction log backup, điểm bắt đầu backup là:

- Điểm kết thúc của transaction log backup trước đó (nếu có một transaction log backup được tạo ra trước đó).
- Transaction log backup như là phần cuối của database backup hoặc differential backup mới nhất nếu không có transaction log backup nào được tạo ra trước đó.

Những trường hợp nên sử dụng transaction log backup:

- Cơ sở dữ liệu quá lớn, đòi hỏi nhiều thời gian và vùng lưu trữ để backup.
- Tính chất quan trọng của hệ thống: những hệ thống đòi hỏi không thể mất mát một dữ liệu nào như ngân hàng, …
- Khôi phục dữ liệu đến một thời điểm bất kỳ.

Để thực hiện sao lưu tập tin transaction log của cơ sở dữ liệu QLSV ta làm tương tự như cách làm của full backup, chỉ khác ở chỗ tại mục Backup type cần lựa chọn loại Transaction log như hình dưới đây:



Hình 4.10. Hộp thoại chọn hình thức sao lưu Transaction Log

Ngoài cách sao lưu trên, có thể thực hiện bằng lệnh như sau:

```
BACKUP LOG QLSV
TO DISK = 'D:\QLSV_log.TRN'
WITH DESCRIPTION = 'QLSV Log Backup';
```

4.2. Phục hồi cơ sở dữ liệu (Restore Database)

Phục hồi CSDL cũng là một công việc rất quan trọng nhằm trả về nguyên hiện trạng CSDL như lúc bắt đầu sao lưu. Như đã trình bày ở phần trên có nhiều phương thức sao lưu và mỗi phương thức này sẽ có một phương thức phục hồi tương ứng.

Việc phục hồi dữ liệu luôn bắt đầu bằng file backup full. Các differential backup và log backup nếu có đều được tiến hành sau khi đã phục hồi bản full backup.

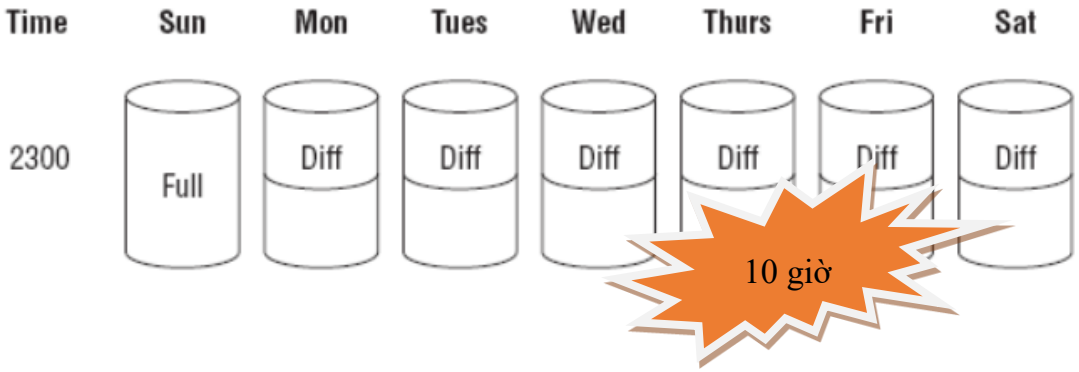
4.2.1. Phục hồi dữ liệu với mô hình Simple Recovery

– Khôi phục bản Full backup sau cùng.

– Khôi phục bản Differential backup mới nhất (nếu có).

Do mô hình Simple Recovery không có backup log nên ta không phục hồi transaction log file.

Ví dụ 2. Giả sử database QLTV với mô hình phục hồi là Simple Recovery có lịch trình backup như hình sau:



Hình 4.11. Lịch trình sao lưu CSDL QLTV

Giả sử có sự cố xảy ra lúc 10 giờ sáng ngày thứ 5, ta tiến hành phục hồi theo thứ tự sau:

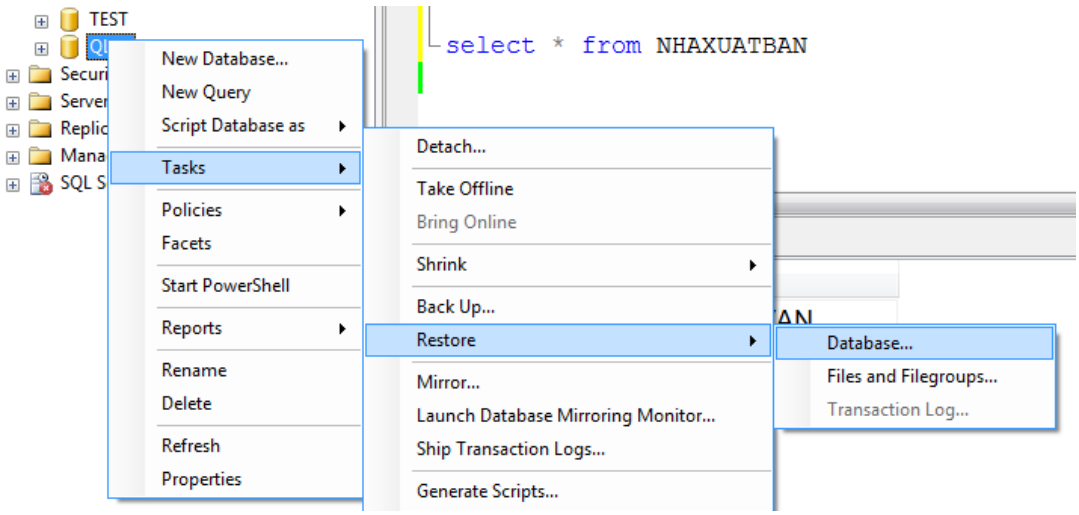
– Phục hồi bản Full backup lúc 23 giờ ngày chủ nhật.

– Phục hồi bản Differential backup lúc 23 giờ ngày thứ 4.

Cách thức thực hiện phục hồi lại CSDL như sau:

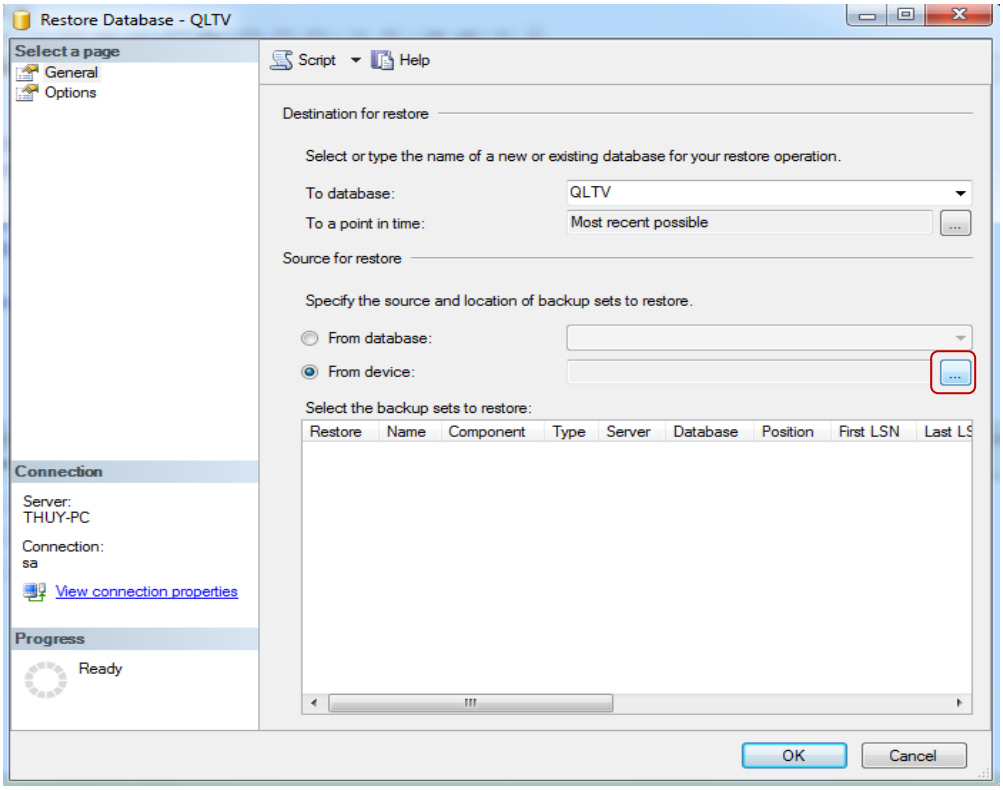
Bước 1: Phục hồi bản Full backup lúc 23 giờ ngày chủ nhật

Click chuột phải vào cơ sở dữ liệu cần phục hồi → chọn Tasks → chọn Restore → chọn Database



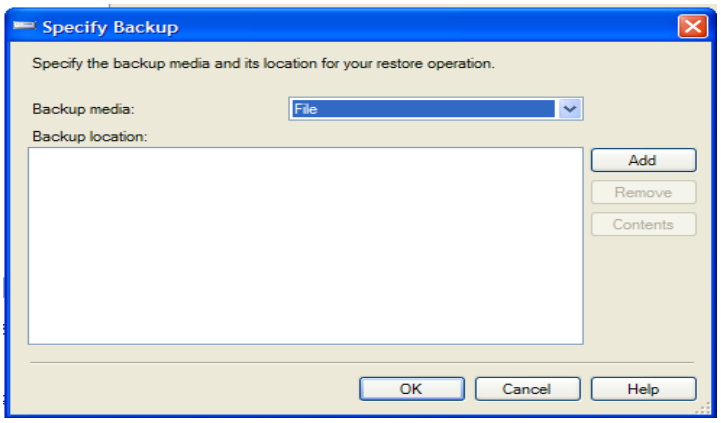
Hình 4.12. Thiết lập lệnh phục hồi 1 CSDL

Trên cửa sổ Restore Database chọn From Device → Chọn ...



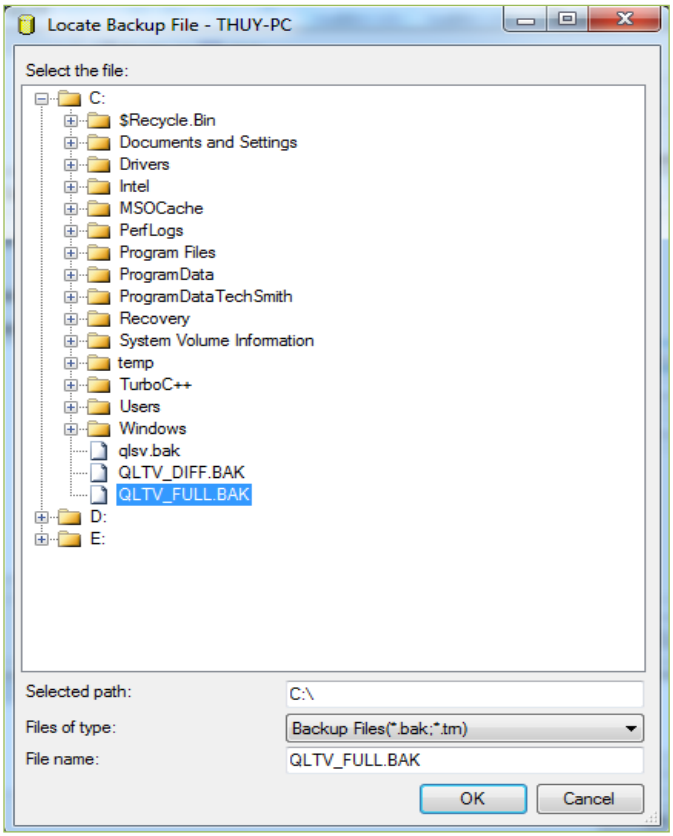
Hình 4.13. Hộp thoại thiết lập thông số khi phục hồi

Khi đó sẽ xuất hiện cửa sổ như hình sau:



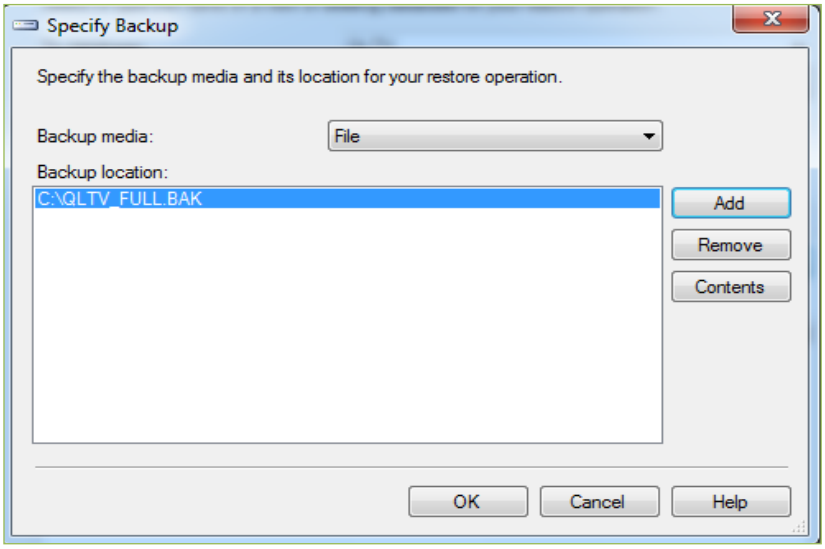
Hình 4.14. Hộp thoại chọn file đã sao lưu

Nhấn nút Add để xuất hiện cửa sổ chọn đường dẫn đến file lưu trữ cơ sở dữ liệu cần phục hồi

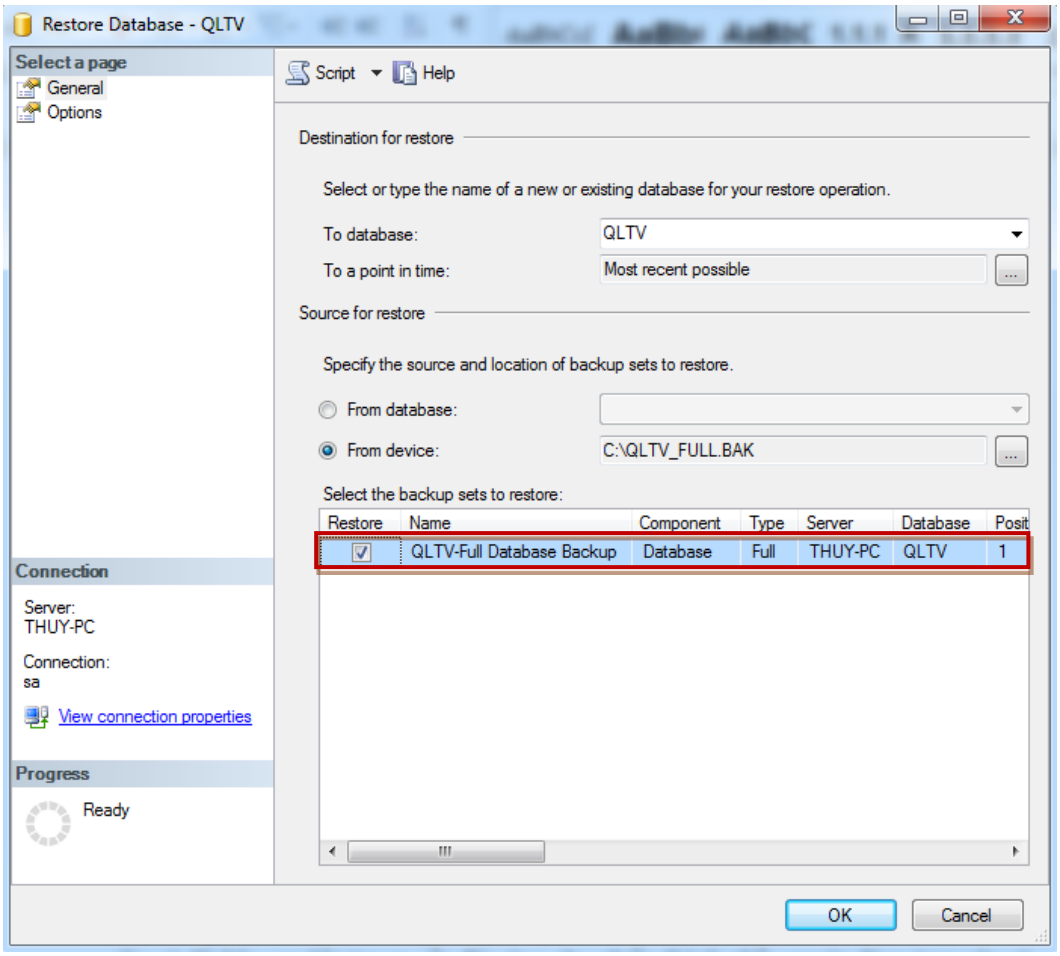


Hình 4.15. Hộp thoại chọn vị trí và tên file đã sao lưu

Click nút OK, xuất hiện hình sau:

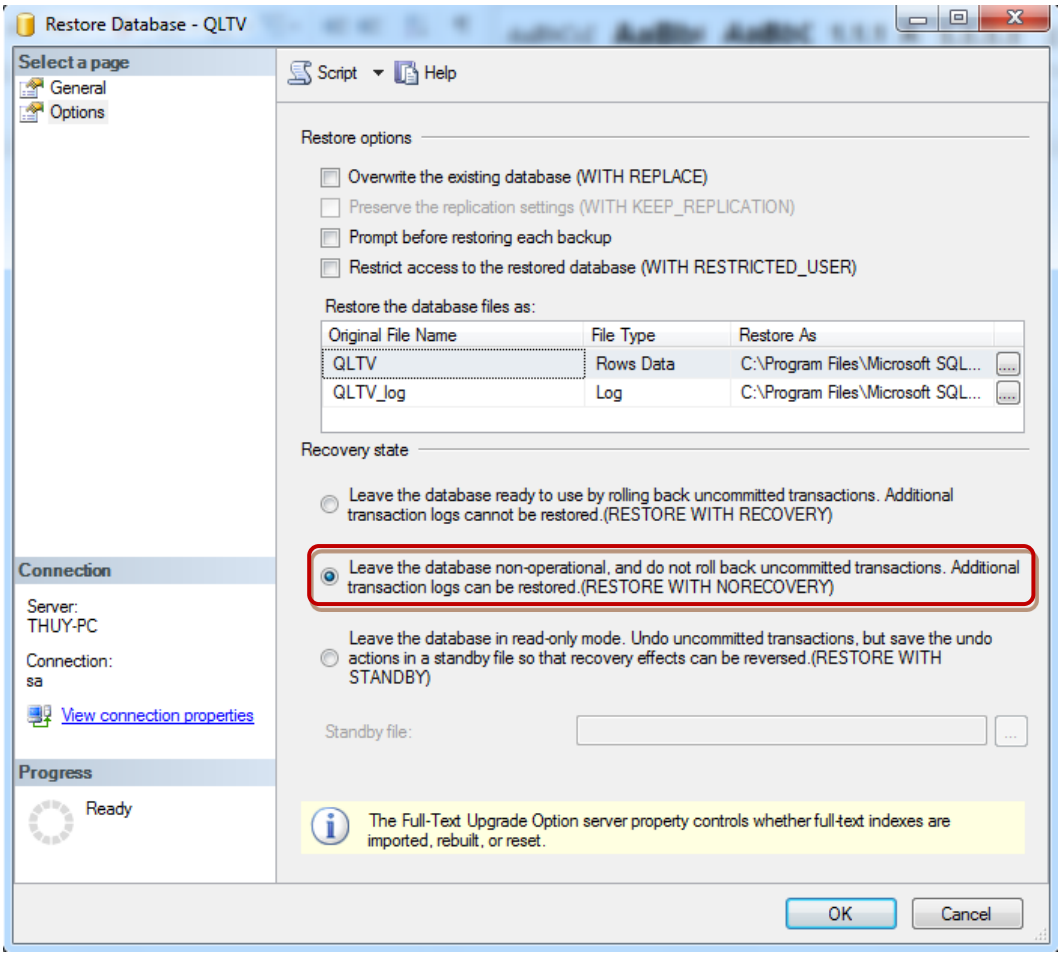


Click nút OK, quay trở lại cửa sổ ban đầu, click chọn file cần restore như hình sau:

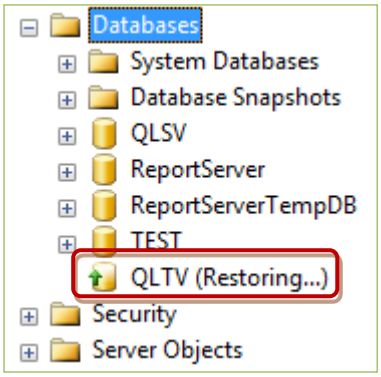


Nếu lúc này nhấn vào nút OK thì quá trình phục hồi kết thúc, ta không thể phục hồi bản Differential backup tiếp theo, dữ liệu chỉ có thể phục hồi tới thời điểm 23 giờ ngày chủ nhật. Nếu muốn phục hồi bản Differential backup tiếp theo, trước khi nhấn nút OK, ta click vào tùy chọn Option, chọn tùy chọn thứ 2: **Leave database non-operational and do not roll back uncommitted transactions. Additional**

transaction logs can be restored. (RESTORE WITH NORECOVERY). Tùy chọn này đặt database ở chế độ chờ phục hồi các bản backup tiếp theo.

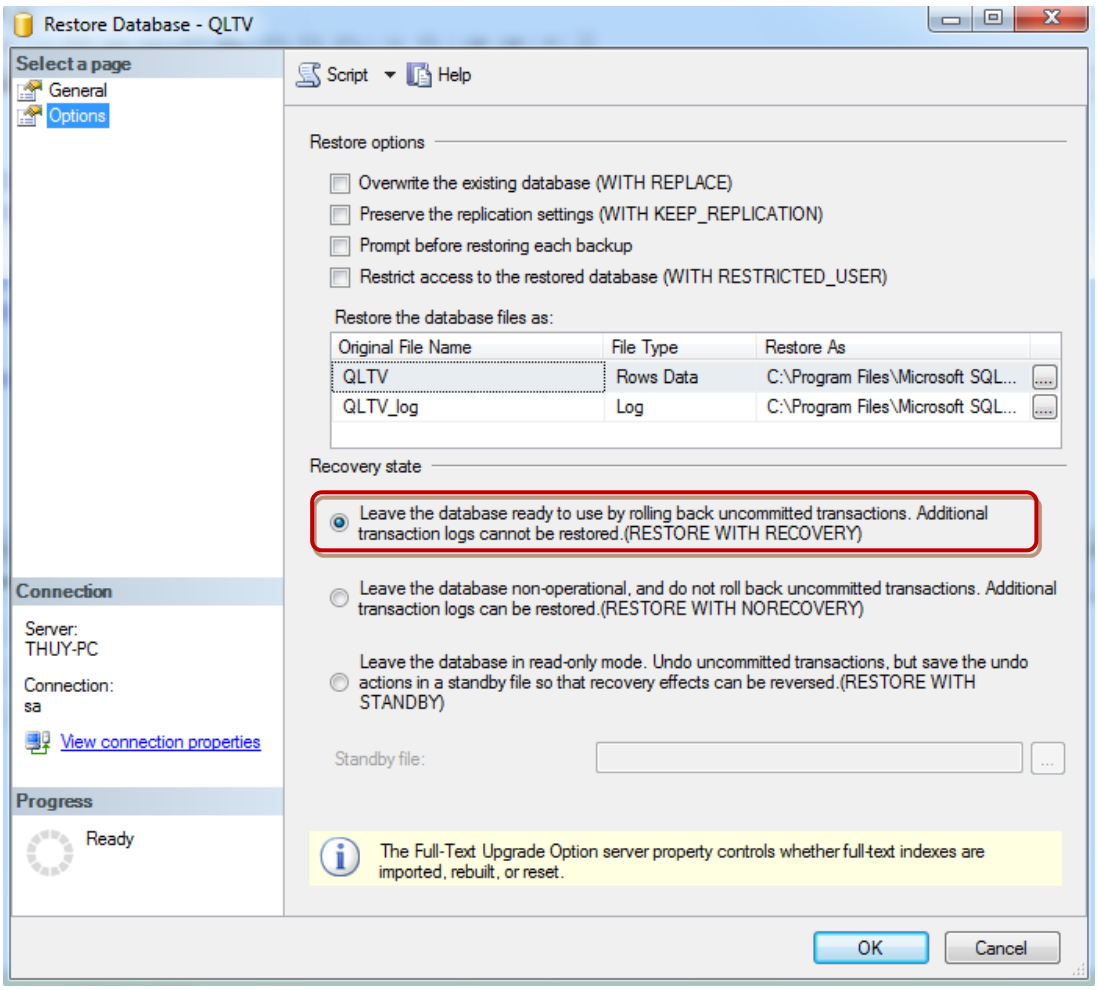


Cuối cùng click vào nút OK, quá trình Restore bản Full backup kết thúc. Tuy nhiên, database ở trạng thái không thể sử dụng được và chờ phục hồi các bản tiếp theo như hình dưới đây:



Bước 2: Phục hồi bản Differential backup lúc 23 giờ ngày thứ 4

Tương tự như phục hồi bản Full Backup. Tuy nhiên, do đây là bản phục hồi sau cùng nên ta chọn tùy chọn đầu tiên để kết thúc quá trình phục hồi như hình sau:



Click nút OK để kết thúc.

Ngoài ra có thể thực hiện phục hồi bằng lệnh như sau:

--bước 1: Phục hồi file backup full gần nhất

```
restore database QLTV
from disk ='C:\QLTV_FULL.bak'
with norecovery
```

/* dùng lựa chọn "WITH NORCOVERY" để sau mỗi lệnh RESTORE sẽ đặt database ở chế độ chờ, tiếp nhận thêm các bản backup tiếp theo (lúc đó database chưa cho phép query). Lệnh RESTORE cuối cùng không dùng lựa chọn này để chỉ ra rằng việc khôi phục đã kết thúc và database đã sẵn sàng hoạt động. */

--bước 2: phục hồi bản different backup gần nhất

```
restore database QLTV
from disk ='C:\QLTV_DIFF.bak'
with recovery
```

Nhận xét: Rõ ràng với mô hình phục hồi Simple Recovery, trong ví dụ này, ta chỉ có thể phục hồi dữ liệu tới thời điểm 23 giờ tối thứ 4, các dữ liệu trong khoảng từ 23 giờ tối thứ 4 đến 10 giờ sáng thứ 5 đã bị mất mà không thể phục hồi lại được.

Do đó, để có thể phục hồi dữ liệu tại bất kỳ một thời điểm nào, ta sử dụng mô hình phục hồi Full Recovery.

4.2.2. Phục hồi dữ liệu với mô hình Full Recovery

Bước 1: Backup "the tail of the log" (cái đuôi), là một dạng backup transaction log. Mục đích của backup cái đuôi là sao lưu các transaction log kể từ lần backup gần nhất tới thời điểm xảy ra sự cố.

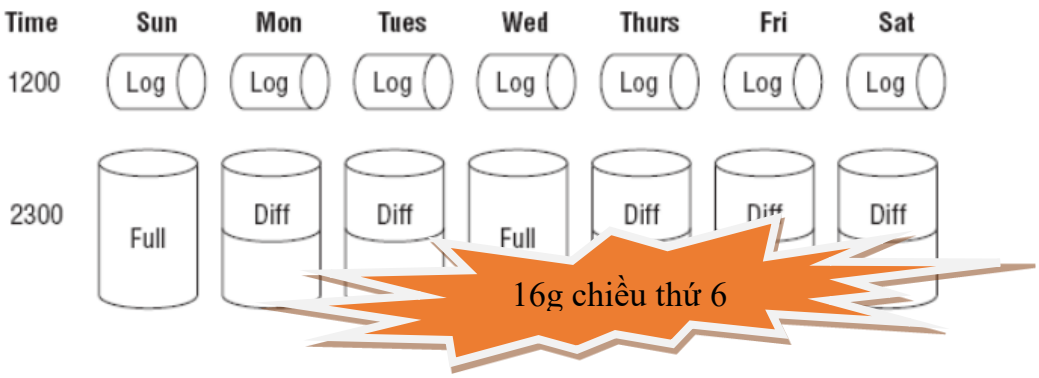
Khi backup "cái đuôi" này cần phải dùng option **NO_TRUNCATE** bởi vì thông thường các Transaction Log Backup sẽ truncate (xoá) những phần không cần dùng đến trong transaction log file, đó là những transaction đã được commited và đã được viết vào database (còn gọi là inactive portion of the transaction log) để giảm kích thước của log file. Tuy nhiên, khi backup phần đuôi, ta không được truncate để đảm bảo tính consistent (nhất quán) của database.

Bước 2: Khôi phục từ bản full backup gần với thời điểm có sự cố nhất. Nó sẽ copy data, log, index... từ đĩa backup vào Data Files và sau đó sẽ lần lượt thực thi các transaction trong transaction log. Lưu ý ta phải dùng option **WITH NORECOVERY** trong trường hợp này (tức là option thứ 2 "**Leave database non-operational and do not roll back uncommitted transactions. Additional transaction logs can be restored. (**RESTORE WITH NORECOVERY)" trong Management Studio). Nghĩa là các transaction chưa hoàn tất (incomplete transaction) sẽ không được roll back. Như vậy database lúc này sẽ ở trong tình trạng không nhất quán và không thể dùng được. Nếu ta chọn **WITH RECOVERY** thì các transaction chưa hoàn tất sẽ được roll back và database ở trạng thái nhất quán nhưng ta không thể nào restore các bản backup khác được nữa.

Bước 3: Khôi phục bản differential backup gần với thời điểm có sự cố nhất (nếu có).

Bước 4: Khôi phục theo thứ tự tất cả các transaction log kể từ lần Full backup (nếu không có bản Differential backup) hay Differential backup sau cùng.

Ví dụ 3. Database QLTV với mô hình phục hồi là Full Recovery có lịch trình backup như hình sau:



Hình 4.16. Lịch trình sao lưu CSDL QLTV

Giả sử sự cố xảy ra lúc 16 giờ chiều thứ 6 làm cho dữ liệu bị mất.

Ta đã có các file backup sau:

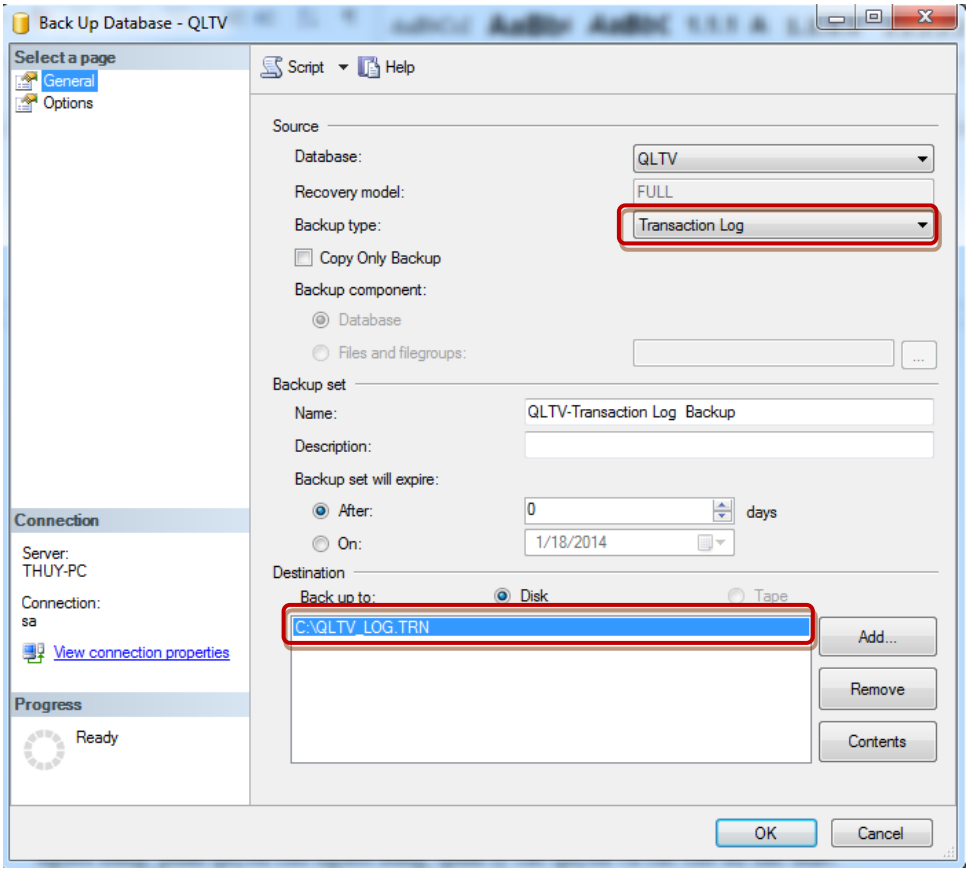
– QLTV_FULL.BAK: file full backup tại thời điểm 23 giờ tối thứ 4.

– QLTV_DIFF.BAK: file differential backup tại thời điểm 23 giờ tối thứ 5.

– QLTV_LOG.TRN: file log backup tại thời điểm 12 giờ trưa thứ 6.

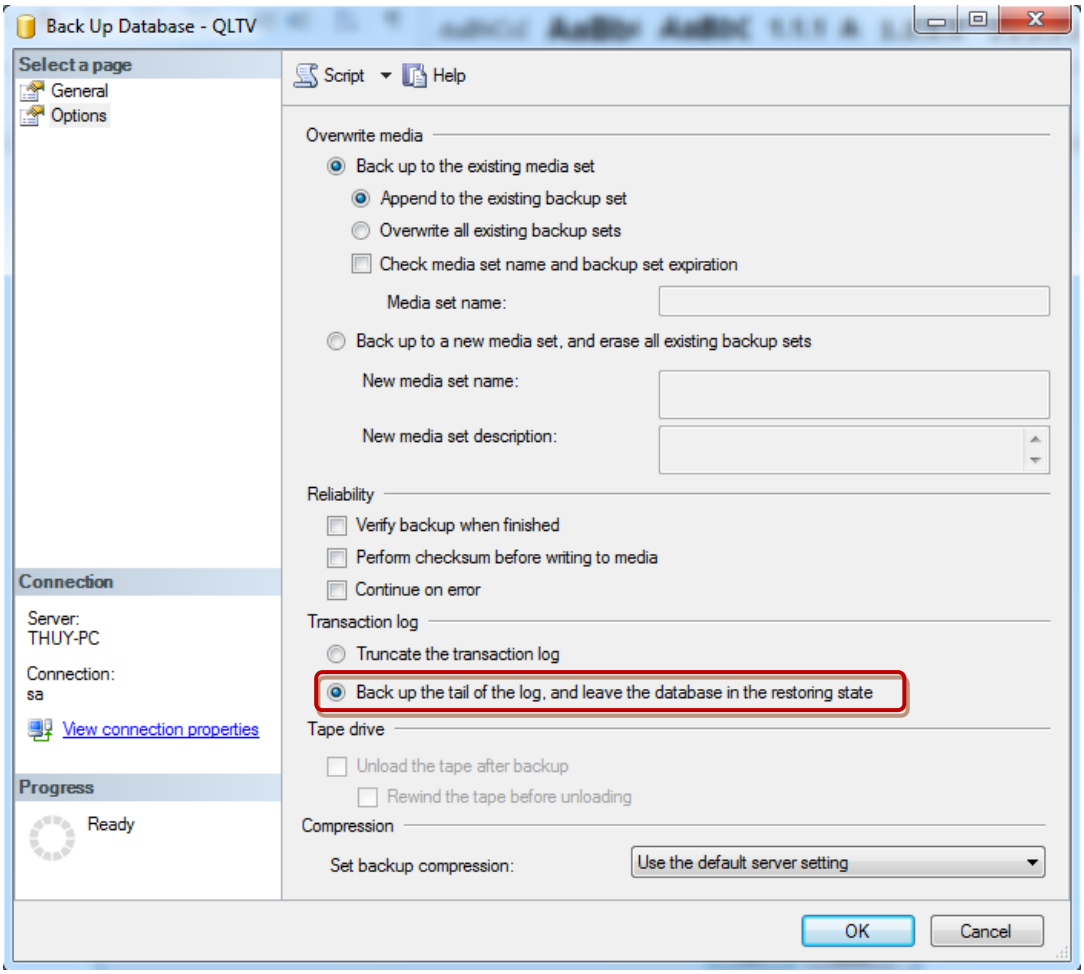
Để phục hồi dữ liệu tới thời điểm xảy ra sự cố, ta thực hiện tuần tự các bước sau:

Bước 1: Backup cái đuôi vào chung file QLTV_LOG.TRN như sau:



Hình 4.17. Sao lưu "cái đuôi" của CSDL QLTV

Vào tùy chọn Option, chọn "**Backup the tail of the log and leave the database in the restoring state**" như hình dưới đây:



Click nút OK, CSDL ở trạng thái Restoring, chờ phục hồi các bản backup tiếp theo.

Bước 2: phục hồi bản Full backup lúc 23 giờ tối thứ 4, với tùy chọn WITH NORECOVERY.

Bước 3: phục hồi bản Differential backup lúc 23 giờ tối thứ 5, với tùy chọn WITH NORECOVERY.

Bước 4: phục hồi bản Log backup lúc 12 giờ trưa ngày thứ 6, với tùy chọn WITH NORECOVERY.

Bước 5: phục hồi bản Log backup cái đuôi, với tùy chọn WITH RECOVERY.

Cách thực hiện giống ví dụ ở phần 4.2.1

Ta cũng có thể phục hồi bằng lệnh như sau:

```
--Bước 1: Backup cái đuôi,lưu những thay đổi từ lúc 12 giờ trưa thứ 6 đến lúc xảy ra sự cố
BACKUP LOG QLTV
```

```
TO DISK= 'C:\QLTV_LOG.trn'
WITH NO_TRUNCATE
```

--Bước 2: Phục hồi file backup full gần nhất (lúc 23 giờ tối thứ 4)

```
restore database QLTV
from disk ='C:\QLTV_FULL.bak'
with norecovery
```

--Bước 3: Phục hồi bản differential backup gần nhất (lúc 23 giờ tối thứ 5)

```
restore database QLTV
from disk ='C:\QLTV_DIFF.bak'
with norecovery
```

--Bước 4: Phục hồi các log backup từ sau lần differential backup gần nhất

--Phục hồi tập tin thứ 1 trong file QLTV_LOG.TRN, chính là bản backup log vào lúc 12 giờ trưa thứ 6

```
restore database QLTV
from disk ='C:\QLTV_LOG.TRN'
with file = 1, norecovery
```

--Phục hồi tập tin thứ 2 trong file QLTV_LOG.TRN, chính là cái đuôi đã backup trong bước 1

```
restore database QLTV
from disk ='C:\QLTV_LOG.TRN'
with file = 2, recovery
```

/*lựa chọn "WITH FILE = x": do khi backup transaction log không có "WITH INIT" nên các bản backup sẽ được nối tiếp vào nhau trong cùng một file "QLTV_LOG.TRN", bản đầu tiên có ID = 1, bản thứ hai có ID = 2 … Do đó khi restore theo thứ tự thời gian, cần chỉ định WITH FILE = 1, 2… */

4.3. Quản lý người dùng và bảo mật hệ thống

Bên cạnh việc sao lưu và phục hồi CSDL thì việc quản lý người dùng và bảo mật hệ thống là một trong những tác vụ quan trọng hàng đầu đối với người quản trị CSDL. Trong phần này chúng ta sẽ tìm hiểu cách tạo và quản lý tài khoản đăng nhập cho người dùng, phân quyền cho người dùng, quản lý các quyền và các chế độ xác thực.

Tại sao phải tạo người dùng đăng nhập?

Cơ sở dữ liệu sau khi tạo ra cần đáp ứng cho nhiều thành phần người dùng truy cập sử dụng, để một cơ sở dữ liệu hoạt động tốt thì cần thiết phải tạo các người dùng đăng nhập để bảo vệ dữ liệu khỏi bị sửa đổi do cố ý hay vô ý của những người dùng không được phép. Khi người dùng đăng nhập vào cơ sở dữ liệu, hệ thống Microsoft SQL Server sẽ nhận biết được từng người dùng với những quyền trên cơ sở dữ liệu của người dùng đó. Mỗi người dùng đăng nhập sẽ được gán một tên duy nhất và một mật khẩu và có một tài khoản đăng nhập riêng.

4.3.1. Các chế độ xác thực

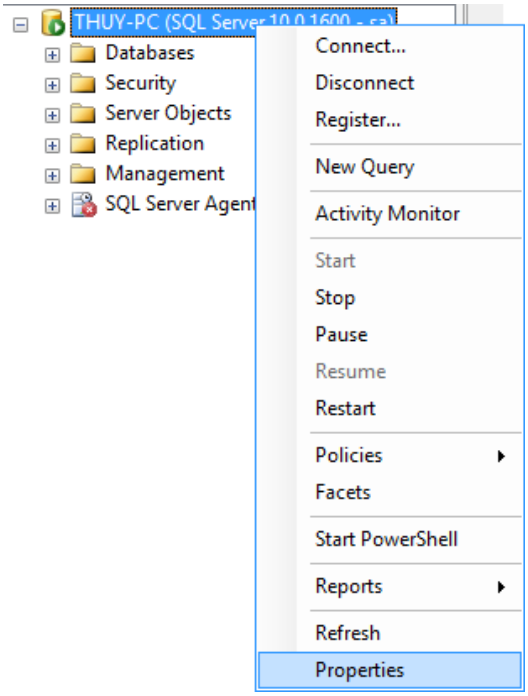
SQL Server xác thực các đăng nhập bằng hai cơ chế, Windows authentication và SQL Server authentication. Việc này bắt nguồn từ khi tạo login trong SQL Server. Khi tạo một login mới, SQL Server sẽ yêu cầu chọn cơ chế xác thực Windows hay SQL Server cho login đó. Nếu chọn Windows, cần cung cấp một windows account và SQL Server sẽ chỉ lưu tên của account đó trong danh sách login. Nếu chọn SQL Server authentication, cần cung cấp login name và password và cả hai đều được lưu trong SQL Server.

Khi đăng nhập vào SQL Server, ta cũng sẽ phải chọn một trong hai cơ chế xác thực. Nếu chọn windows, chính account hiện đang đăng nhập vào windows được dùng. Chúng ta không có quyền chọn login, SQL Server sẽ kiểm tra xem account nào đang đăng nhập vào windows, sau đó xem account đó có nằm trong danh sách login của SQL hay không. Nếu có thì cho login vào còn nếu không thì sẽ chặn lại.

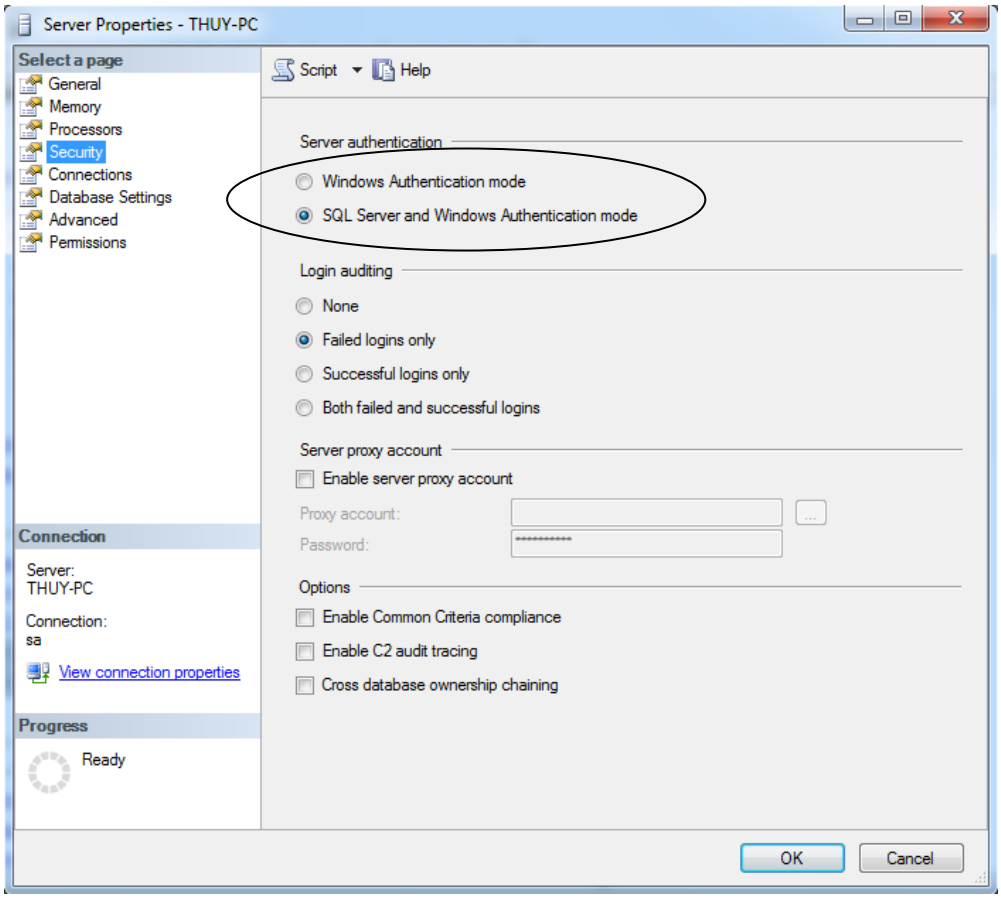
Còn với SQL Server Authentication thì cần cung cấp login ID và password. SQL Server sẽ kiểm tra và quyết định có cho login vào hay không.

Chế độ xác thực được chọn trong quá trình cài đặt hệ quản trị SQL Server 2008, nếu muốn thay đổi, thực hiện như sau:

Trong cửa sổ Object Explorer nhấp chuột phải lên Server → chọn Properties



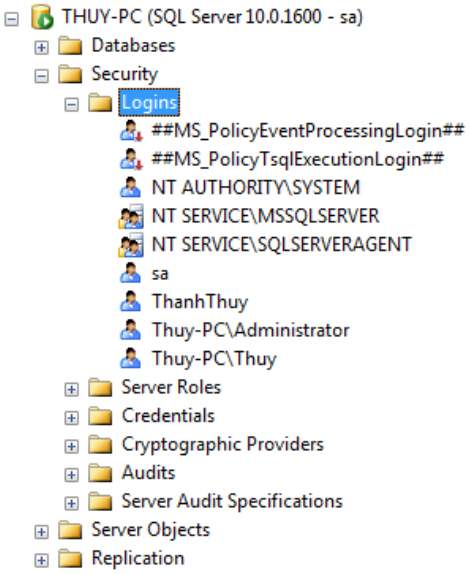
Trên cửa sổ Server Properies mở trang Security → chọn kiểu xác thực → chọn OK



Hình 4.18. Hộp thoại Server Properies – Thiếp lập quyền xác thực

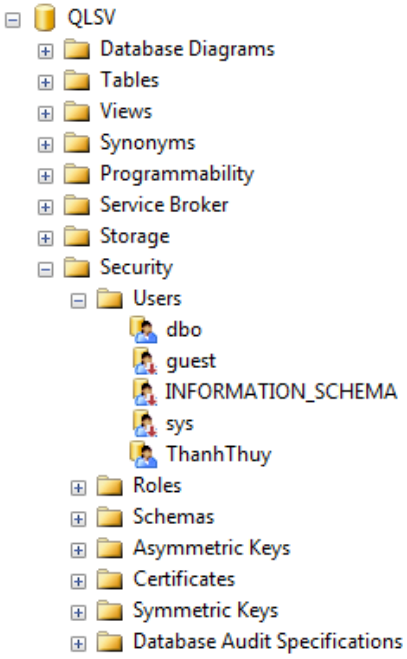
4.3.2. Login và User

SQL Server cho phép truy nhập vào hệ thống thông qua các login. Chỉ khi có quyền ở mức độ nhất định thì mới có thể tạo được login. Danh sách các login nằm ở mục Security/Logins như hình sau:



Các login này mới chỉ có quyền truy nhập vào server chứ chưa hẳn đã truy nhập được vào các database chứa trong đó.

Muốn truy cập vào các database, ta phải tạo user. Mỗi database có một danh sách các user, các user này luôn luôn gắn (mapped) với một login ở mức server. Khi người dùng đăng nhập vào SQL Server thông qua login này, người dùng sẽ có quyền truy nhập vào database theo quyền hạn mà user tương ứng với nó được cung cấp. Danh sách các user nằm ở mục Security/Users của database tương ứng.



Ví dụ 4. Ta có một login tên là ThanhThuy trong SQL Server. Ở database QLSV ta có user ThanhThuy được gắn với login ThanhThuy, user này có quyền select trên bảng SINHVIEN. Ở database QLTV ta cũng có user ThanhThuy được gắn với cùng login trên, và user này có quyền Insert, Update, Delete trên bảng DOCGIA. Khi truy nhập vào SQL Server bằng login ThanhThuy, ta sẽ có quyền select trên bảng SINHVIEN của database QLSV và quyền Insert, Update, Delete trên bảng DOCGIA của database QLTV.

4.3.2.1. Tài khoản đăng nhập (Login Account)

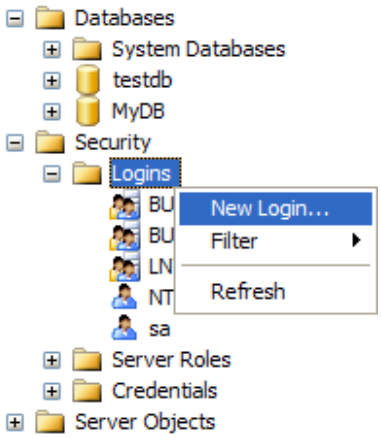
Tài khoản đăng nhập là một tài khoản mà người dùng (user) sử dụng để đăng nhập vào hệ thống SQL Server. Một login có thể có quyền truy cập 0..n database. Trong mỗi database, một login sẽ ứng với một user. Một login được cấp và quản lý bởi sysadmin hoặc security admin.

Có 2 loại tài khoản đăng nhập:

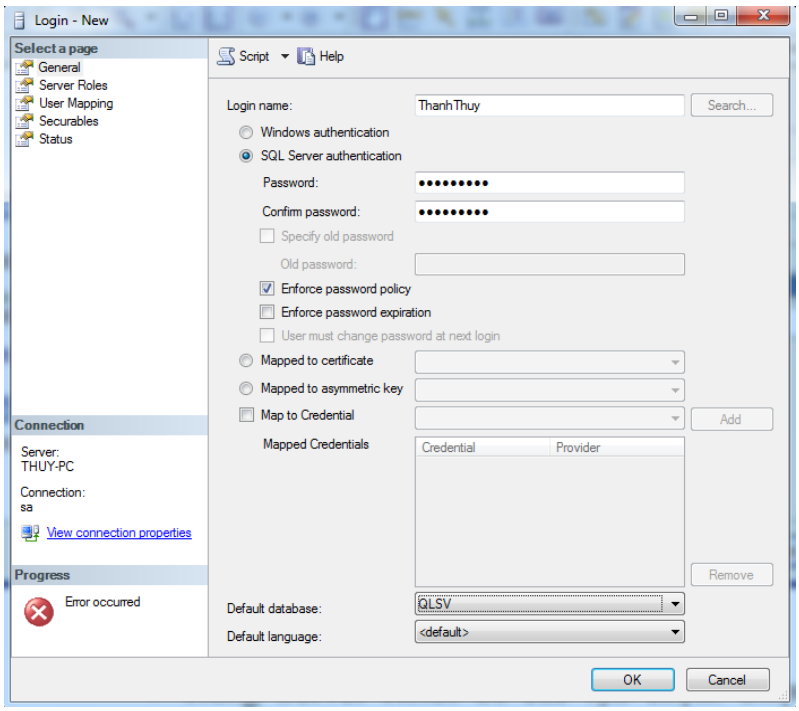
+ SQL Server Login ID
+ Windows Login ID

Để tạo tài khoản đăng nhập ta thực hiện như sau:

Mở rộng danh mục Security trong server hiện hành → nhấp chuột phải vào Logins → chọn New login

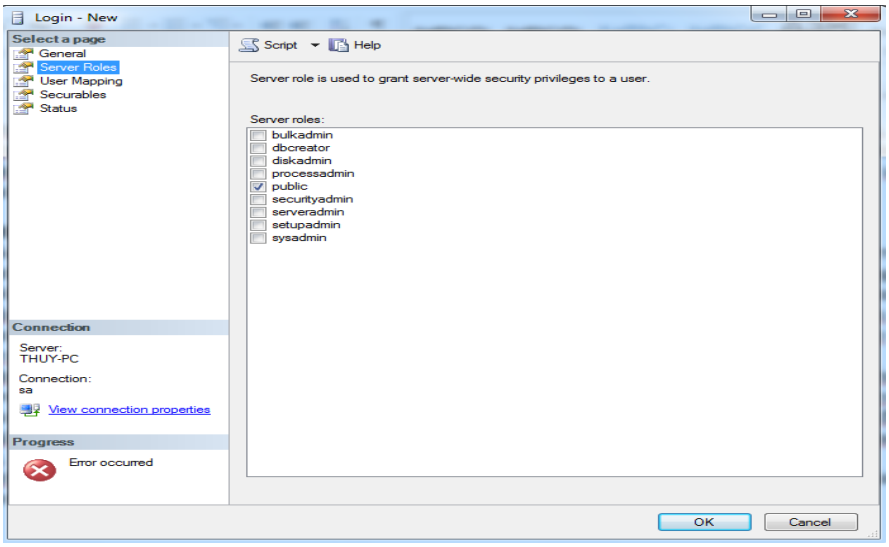


Trên cửa sổ Login, trong phần Select a page chọn General và đặt tên tài khoản đăng nhập trong hộp Login name. Tiếp theo chọn chế độ xác thực SQL Server Authentication. Nếu chọn chế độ xác thực Windows Authentication thì tên tài khoản này phải là một tài khoản đang tồn tại trong hệ điều hành Windows. Trong hộp Password nhập mật khẩu đăng nhập. Trong mục Default database và default language chọn cơ sở dữ liệu và ngôn ngữ mặc định sẽ được dùng.



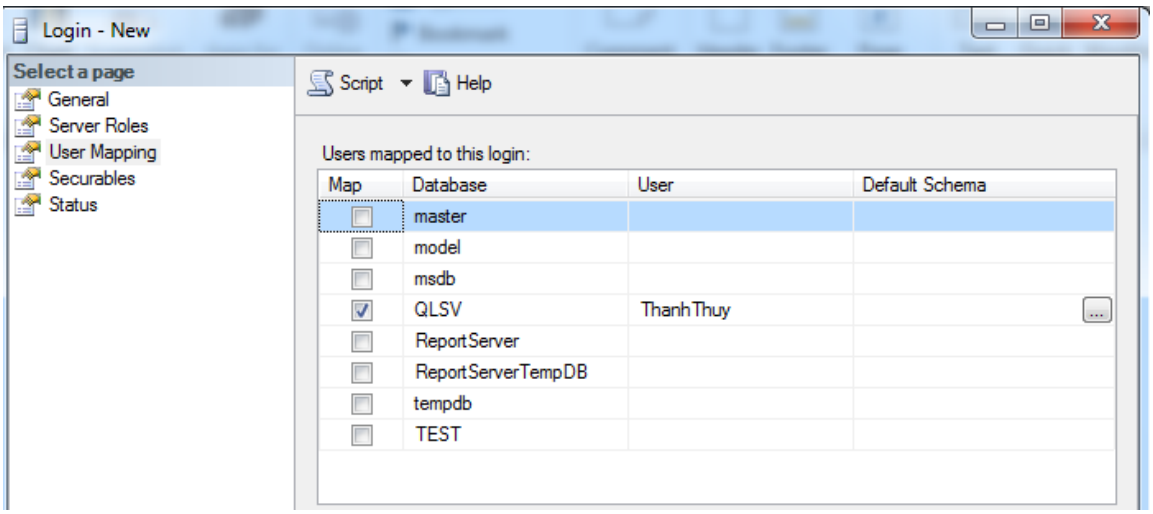
Hình 4.19. Hộp thoại tạo 1 tài khoản đăng nhập

Trong phần Select a page chọn Server Roles (nhóm quyền Server), trong trang này có thể chỉ ra nhóm quyền server cho login mới bằng cách chọn nhóm quyền từ danh sách nhóm quyền. Với những người sử dụng thường thì chúng ta có thể không cấp các nhóm quyền này. Chức năng cụ thể của từng nhóm quyền server sẽ được trình bày ở phần sau.



Hình 4.20. Hộp thoại thiết lập Server Roles cho tài khoản đăng nhập

Click chọn mục User mapping, trên trang này chúng ta có thể chỉ định cơ sở dữ liệu mà người dùng được phép truy cập vào bằng cách chọn vào ô kiểm trong cột map, khi đó tại cột user của database sẽ tự động tạo ra một user có tên mặc định cùng với tên tài khoản đăng nhập (có thể thay đổi thành tên khác).



Hình 4.21. Hộp thoại thiết lập CSDL truy cập cho tài khoản đăng nhập

Click chọn OK để kết thúc quá trình tạo tài khoản đăng nhập.

Ngoài ra chúng ta có thể tạo tài khoản đăng nhập bằng lệnh như sau:

– **Tạo Windows login ID bằng lệnh:**

```
SP_GRANTLOGIN  <Windows_Account>
```

hoặc

```
CREATE LOGIN < Windows_Account > FROM WINDOWS
```

✍ **Lưu ý**: <Windows_Account> phải là một account của Windows, có dạng Domain\User

Sau khi được cấp quyền, tài khoản Windows được sử dụng như một login của SQL Server.

Ví dụ 5. `sp_grantlogin 'Thuy-PC\Thuy'`

Hoặc `create login [Thuy-PC\Thuy] from windows`

– **Tạo SQL Server login ID**

```
sp_addlogin <tên đăng nhập>,
            <mật khẩu>
           [,<CSDL mặc định>]
```

hoặc

```
Create Login < tên đăng nhập>
with Password = <password>[,
default database = <CSDL mặc định>]
```

Trong đó:

– Tên đăng nhập: Là tên của tài khoản được dùng để đăng nhập vào hệ thống.

– Mật khẩu: Là mật khẩu đăng nhập vào hệ thống

– CSDL mặc định: Là CSDL sẽ được truy cập vào khi đăng nhập bằng tên tài khoản và mật khẩu đã chỉ định. Nếu không cung cấp CSDL mặc định thì CSDL Master sẽ được chọn làm CSDL mặc định.

Ví dụ 6. Tạo tài khoản đăng nhập với tên là ThanhThuy, mật khẩu là 123456, CSDL mặc định là QLSV.

```
sp_addlogin 'ThanhThuy' , '123456' , 'QLSV'
```

hoặc

```
create login ThanhThuy
with password = '123456', default_database = QLSV
```

Để huỷ tài khoản đăng nhập có thể thực hiện bằng cách click chuột phải vào tài khoản đăng nhập → chọn delete → chọn OK. Hoặc có thể sử dụng cú pháp như sau:

```
sp_droplogin <tên_tài_khoản_đăng_nhập>
```

hoặc

```
drop login <tên_tài_khoản_đăng_nhập>
```

Ví dụ 7. `sp_droplogin 'ThanhThuy'`

Hoặc `drop login ThanhThuy`

4.3.2.2. Tạo tài khoản người dùng (User Account)

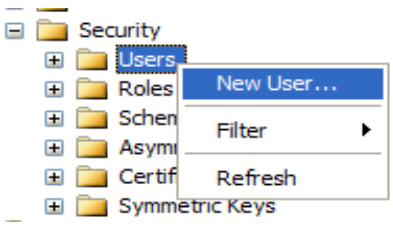
Tạo user chính là cấp cho một login quyền truy cập vào database hiện hành.

Để tạo một user, trước hết phải tạo một login cho user đó bởi vì tên user sẽ ánh xạ (mapping) đến login.

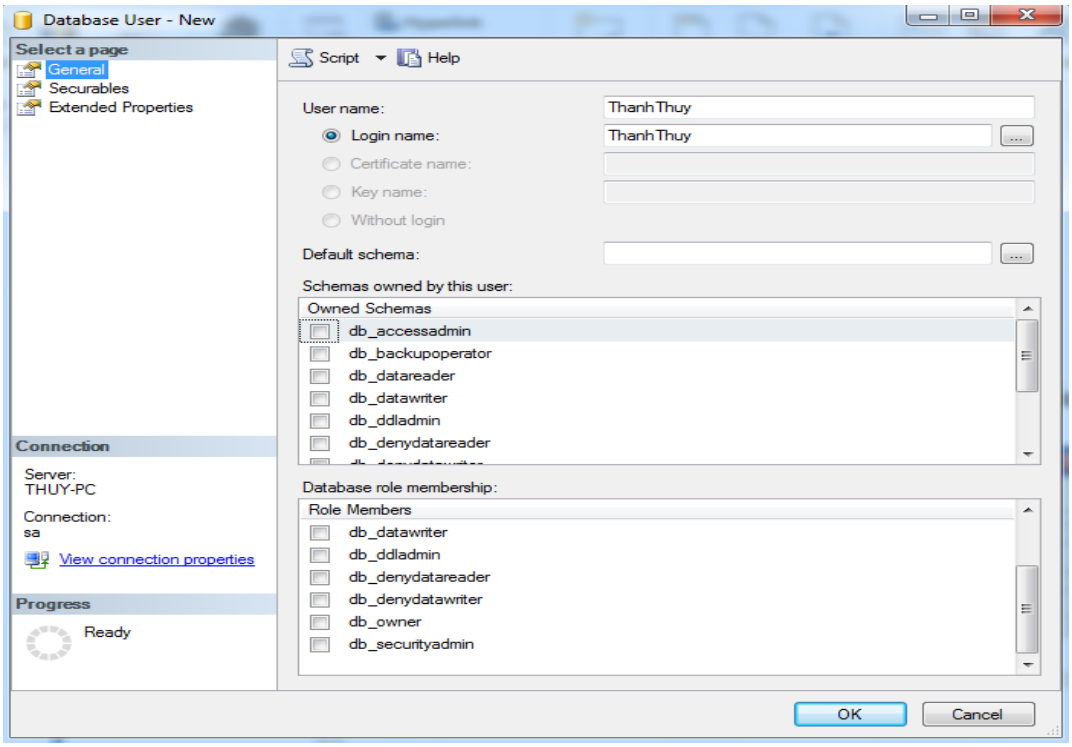
Không giống như tên đăng nhập được tạo từ danh mục Security của server, tên người dùng được tạo từ trong danh mục cơ sở dữ liệu cụ thể.

Để tạo tài khoản người dùng, thực hiện như sau:

Mở rộng mục Security trong cơ sở dữ liệu hiện hành → nhấp chuột phải vào đối tượng Users → chọn New User.



Trên cửa sổ Database User gõ tên user và chọn một tài khoản đăng nhập tương ứng (tài khoản đăng nhập được chọn phải là tài khoản chưa được cấp cho user nào)



Hình 4.22. Hộp thoại tạo tài khoản người dùng

Nhấp chọn OK để kết thúc quá trình tạo tài khoản người dùng.

Ngoài ra chúng ta có thể sử dụng cú pháp như sau:

```
sp_grantdbaccess <tênđăngnhập>[,<tên người dùng>]
```

hoặc

```
sp_adduser <tên đăng nhập>,<tên người dùng>
```

hoặc

```
create user <tên người dùng>
for login <tên đăng nhập>
```

Trong đó:

- Tên đăng nhập: Là tên tài khoản đăng nhập đã được tạo trước đó.
- Tên người dùng: Là tên người dùng cần tạo. Nếu không chỉ ra tên người dùng thì mặc định sẽ cùng tên với tên đăng nhập.

✍ **Lưu ý**:

- Chọn cơ sở dữ liệu ngữ cảnh trước khi thực thi câu lệnh.
- Nếu không xác định tên người dùng, tên người dùng là tên đăng nhập.

- Thủ tục Grantdbaccess chỉ có thể được thực hiện bởi thành viên của sysadmin, db_owner, db_accessadmin.
- Mỗi database có 2 user mặc định:
  + Dbo: là user có tất cả các quyền trên database
    - Tất cả các thành viên thuộc sysadmin server role đều là dbo của tất cả các database trên server.
    - Một thành viên của db_owner database role cũng có quyền như dbo.
  + Guest: được dùng khi login account muốn truy xuất đến một database nhưng không có user account trong database đó, thì có thể đến database như một người khách. Chúng ta cũng có thể cấp quyền cho guest user giống như cấp quyền cho các user account khác.

Ví dụ 8. Tạo người dùng có tên là Thuy trên cơ sở dữ liệu QLSV với tên tài khoản đăng nhập là ThanhThuy

```
use QLSV
go
sp_adduser 'ThanhThuy','Thuy'
```

hoặc

```
create user Thuy for login ThanhThuy
```

Để huỷ tài khoản người dùng có thể thực hiện bằng cách click chuột phải vào tài khoản người dùng → chọn Delete → chọn OK. Hoặc có thể sử dụng thủ tục hệ thống như sau:

```
sp_dropuser  <tên_tài_khoản_người_dùng>
```

Ví dụ 9. Huỷ tài khoản người dùng có tên là Thuy

```
sp_dropuser 'Thuy'
```

4.3.3. Quản lý nhóm quyền trên cơ sở dữ liệu

Trong một cơ sở dữ liệu có nhiều người dùng, và mỗi người dùng đều được cấp một số quyền nhất định. Chúng ta sẽ gặp khó khăn trong việc quản lý quyền và người sử dụng nếu như cơ sở dữ liệu có số lượng người dùng nhiều và các quyền cấp phát đa dạng. Do đó để đơn giản hoá việc quản lý nhiều quyền cấp cho nhiều người dùng, Microsoft SQL Server đưa ra nhóm quyền (roles). Các nhóm quyền được thiết kế cho phép các nhóm người dùng nhận các quyền cơ sở dữ liệu giống nhau mà không phải cấp những quyền này một cách riêng biệt. Thay vì cấp từng quyền cho từng người dùng, chúng ta tạo ra nhóm quyền đại diện cho các quyền được sử dụng bởi một nhóm người dùng và sau đó cấp nó cho nhóm người dùng.

Các nhóm quyền có thể chia thành 2 loại: server roles và database roles

4.3.3.1. Server roles

Server role là nhóm các quyền cố định ở mức server do hệ thống tạo sẵn, người dùng không thể thay đổi được. Những nhóm quyền này được dùng để cấp quyền cho người quản trị cơ sở dữ liệu. Một login khi được cấp một trong các nhóm quyền này sẽ có thể thực hiện được một số thao tác nhất định ở mức server. Nhóm quyền mặc định khi tạo mới một login là quyền public, nhóm quyền này thực chất là không có quyền gì ngoài quyền truy nhập vào server. Thông thường chỉ DBA mới nên có quyền sysadmin, còn các developer chỉ cần quyền public và bổ sung thêm khi cần.

Khi mới cài đặt, SQL Server định nghĩa sẵn login **sa**. Login **sa** và các login là administrator của Windows đều là thành viên của Sysadmin.

Bảng 4. 1. Các nhóm quyền server:

Server Role	Diễn giải
Bulkadmin	Có thể thực thi lệnh BULK INSERT để thêm lượng lớn dữ liệu vào bảng.
Dbcreator	Tạo và chỉnh sửa cơ sở dữ liệu
Diskadmin	Quản lý các tập tin trên đĩa
Processadmin	Quản lý các tiến trình đang chạy trên SQL Server
Securityadmin	Quản lý bảo mật an toàn hệ thống như tạo login, gán quyền …
Serveradmin	Có quyền cấu hình mức server
Setupadmin	Quản lý các thủ tục, server liên kết…
Sysadmin	Quyền tối đa, có thể thực hiện mọi thao tác trên server.

– Để xem các fixed server roles

```
sp_helpsrvrole
```

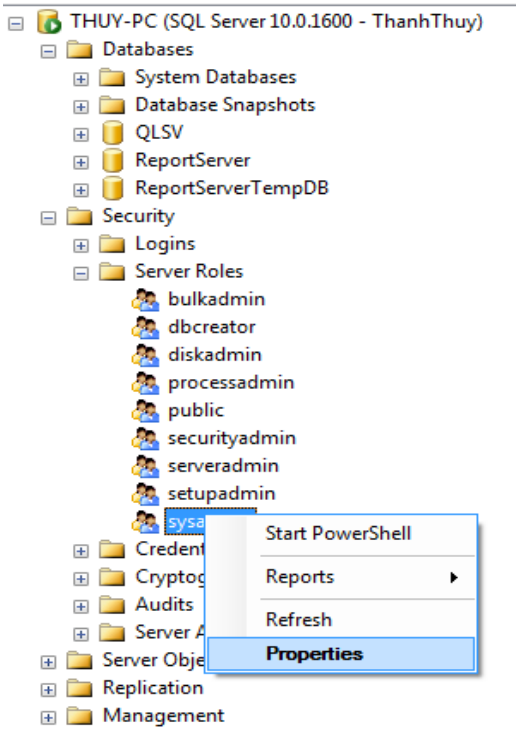
– Để xem chi tiết quyền của mỗi role

```
sp_srvrolepermission
```

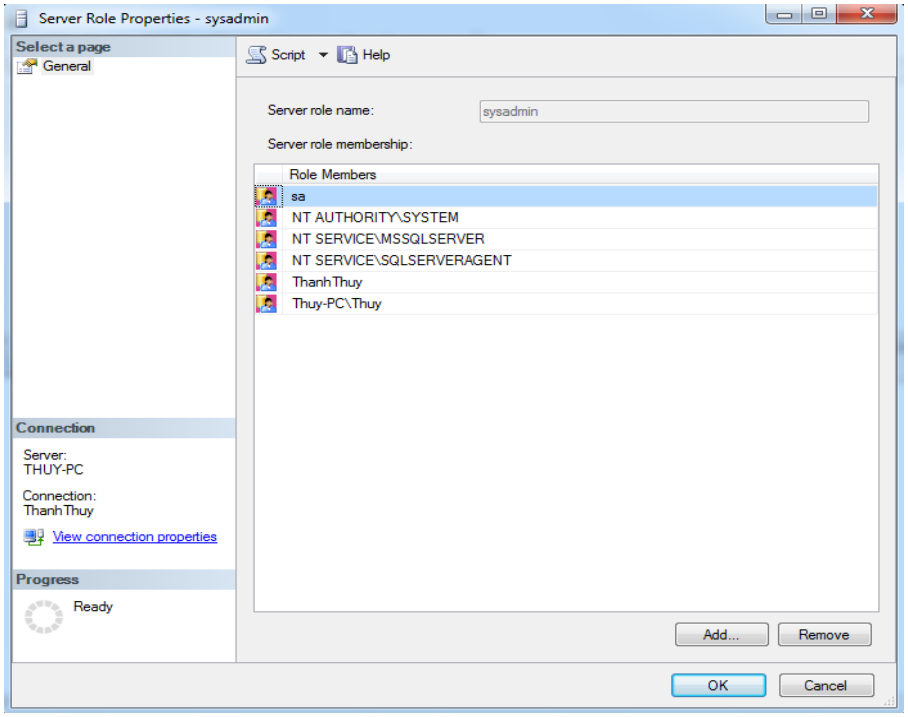
– Để thêm người dùng vào 1 fixed server role

Cách 1: Dùng SQL Server Management Studio:

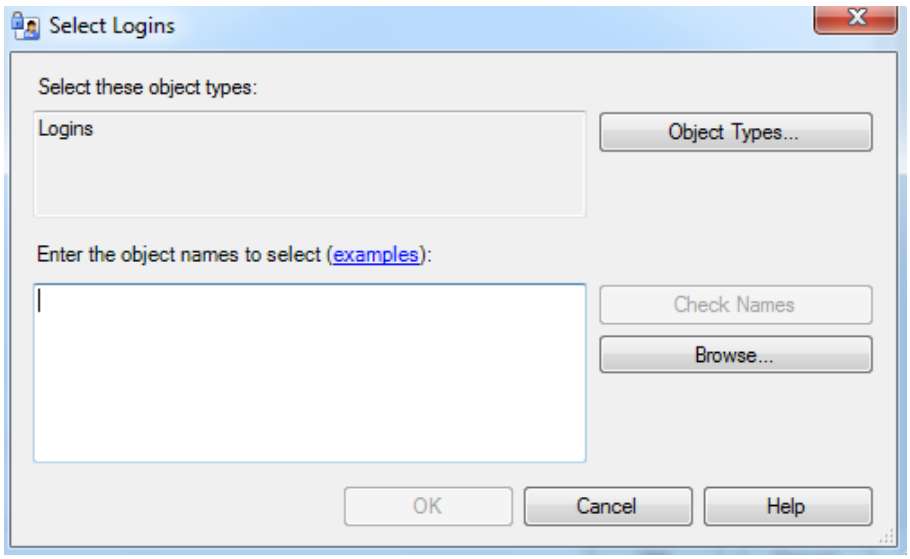
Click chuột phải lên Server Role muốn thêm người dùng → chọn Properties:



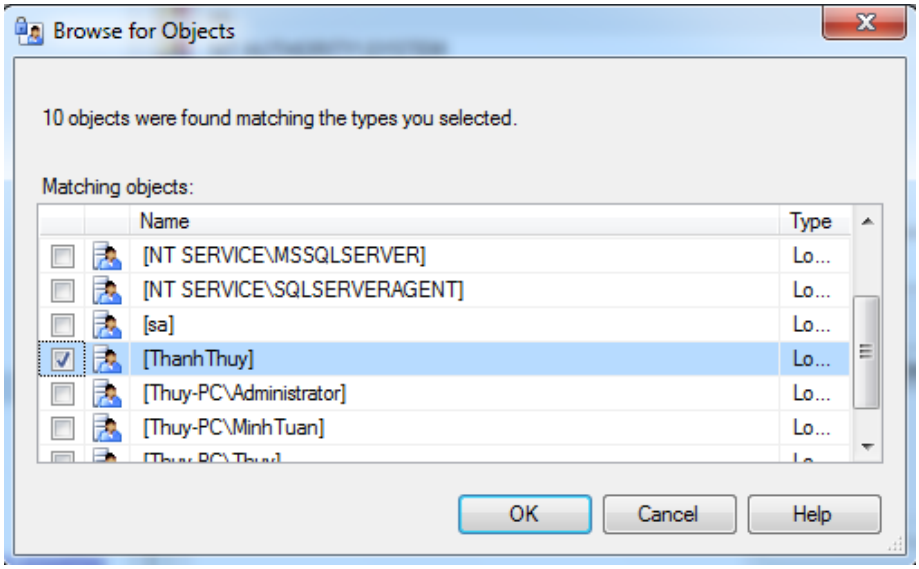
Click vào nút Add trong hộp thoại sau:



Hộp thoại Select Logins xuất hiện:



Click vào nút Browse, xuất hiện hộp thoại Browse For Objects



Chọn các login đưa vào nhóm và click nút OK.

Cách 2: Dùng lệnh:

```
sp_addsrvrolemember 'login', 'role'
```

Ví dụ 10. Gán login 'ThanhThuy' vào nhóm sysadmin

```
sp_addsrvrolemember 'ThanhThuy', 'sysadmin'
```

Ví dụ 11. Thêm user của Windows NT Thuy-PC\Thuy vào role sysadmin

```
sp_addsrvrolemember 'Thuy-PC\Thuy', 'sysadmin'
```

4.3.3.2. Database roles

Database role được sử dụng để cung cấp các mức khác nhau khi truy cập vào cơ sở dữ liệu. Các user khi được thêm vào database đều mặc định có quyền public. Với quyền public, user chỉ có thể nhìn thấy tên database mà không có quyền gì khác.

Database role có 2 loại:

- Fixed database role.
- User defined database role.

a. Fixed database roles

Fixed database role là nhóm các quyền cố định ở mức database do hệ thống tạo sẵn, người dùng không thể thay đổi được, gồm các nhóm quyền trong bảng sau:

Bảng 4. 2. Nhóm quyền cố định ở mức database do hệ thống tạo sẵn:

Database Role	Diễn giải
db_owner	Đây là role cao nhất mà một người dùng có thể có. Cho phép người dùng có mọi quyền trên database.
db_accessadmin	Cung cấp cho người dùng quyền thêm hoặc xóa các người dùng khác trong cơ sở dữ liệu.
db_securityadmin	Cho phép người dùng quản lý các nhóm quyền và quyền trong cơ sở dữ liệu.
db_ddladmin	Cho phép người dùng thực thi các câu lệnh DDL.
db_backupoperator	Cho phép người dùng thực hiện backup dữ liệu.
db_datareader	Cho phép người dùng truy xuất dữ liệu từ tất cả các bảng trong database.
db_datawriter	Cho phép người dùng thêm, xóa, sửa dữ liệu trên tất cả các bảng của database.
db_denydatareader	Người dùng không thể truy xuất dữ liệu từ tất cả các bảng trong database.
db_denydatawriter	Người dùng không thể thêm, xóa, sửa dữ liệu trên toàn bộ các bảng của database.

- Để xem các fixed database role:

```
sp_helpdbfixedrole
```

- Để xem các quyền của mỗi fixed database role:

```
sp_dbfixedrolepermission
```

- Thêm một user vào database role:

```
sp_addrolemember 'database role', 'user'
```

Ví dụ 12. Gán người dùng ThanhThuy vào nhóm quyền db_datareader

```
sp_addrolemember 'db_datareader', 'ThanhThuy'
```

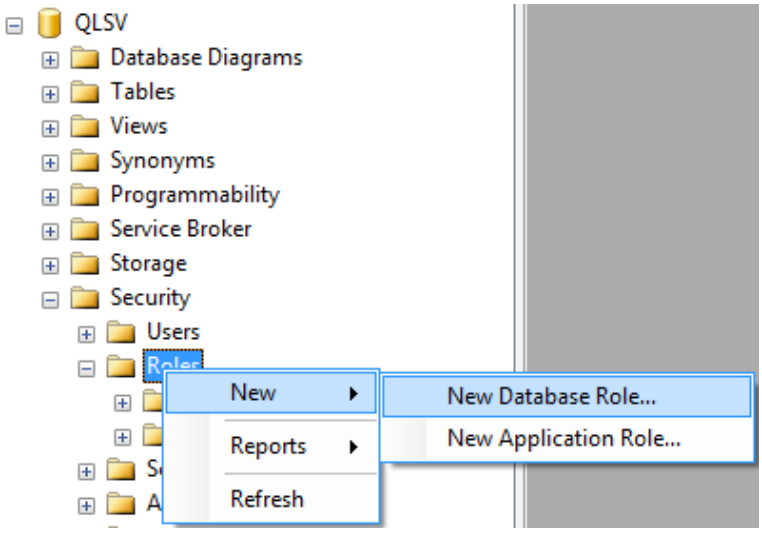
b. User defined database role

Ngoài các nhóm quyền do hệ thống tạo sẵn, người dùng có thể cần có thêm một số quyền mà các quyền này chưa được xác định trong bất kỳ fixed database role nào. SQL server cho phép người dùng tự tạo ra các nhóm quyền riêng gọi là nhóm quyền do người dùng định nghĩa.

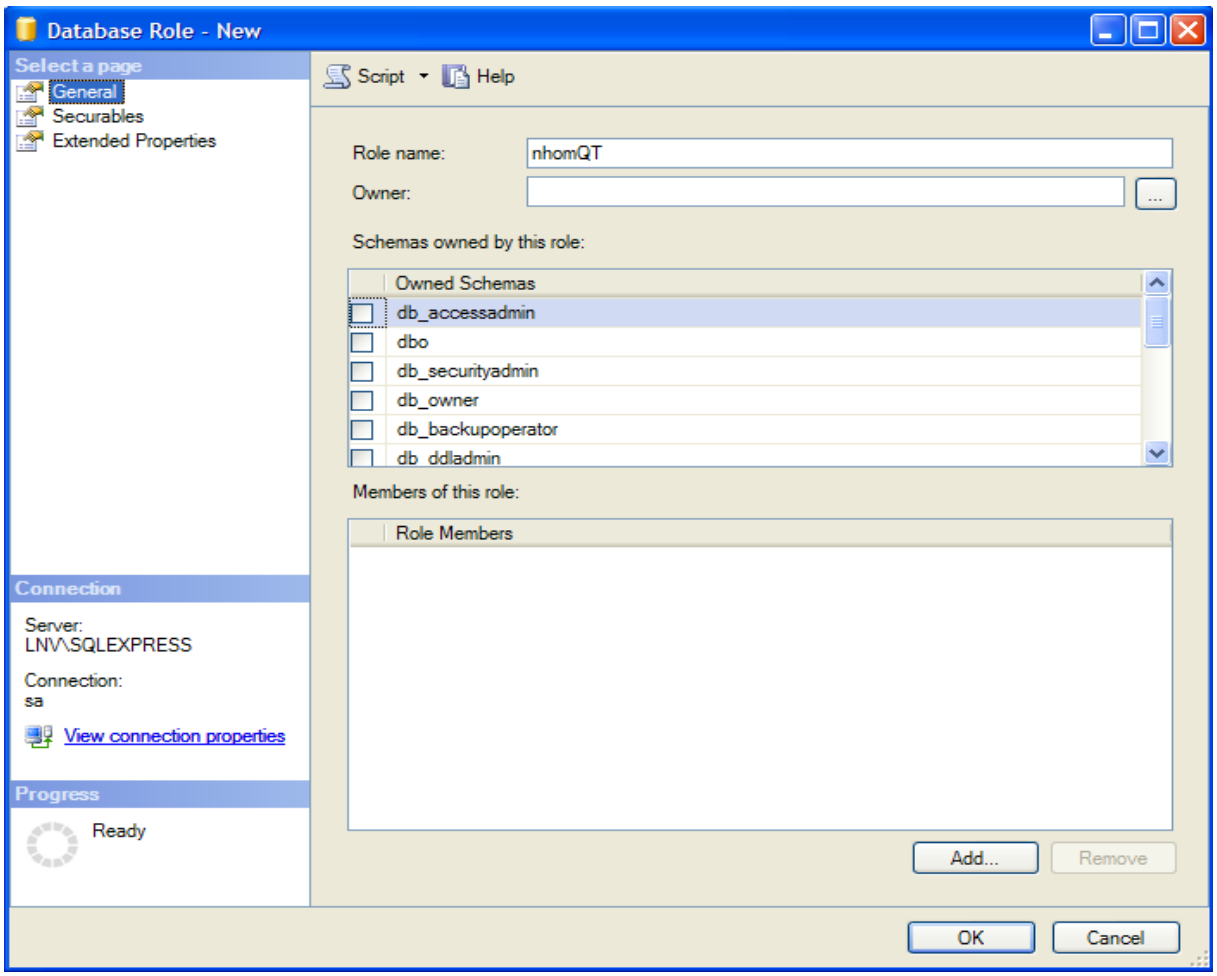
Để sử dụng được nhóm quyền, đầu tiên chúng ta phải tạo ra nhóm quyền, sau đó gán các quyền cho nhóm quyền và thêm người dùng vào nhóm quyền.

– **Tạo mới một nhóm quyền:**

Nhấp chuột phải vào đối tượng Roles → chọn New → Chọn New Database Role



Hộp thoại Database Role xuất hiện:



Trong phần Select a page chọn trang General → Đặt tên Role tại hộp Role name → Chọn trang Sercurables nhấp chọn nút add để thêm quyền vào nhóm quyền. Sau khi cấp quyền xong, chọn lại trang General → nhấn nút add để thêm người dùng vào nhóm quyền.

Chúng ta cũng có thể tạo nhóm quyền bằng lệnh T-SQL như sau:

– **Tạo nhóm quyền:**

```
sp_addrole <tên nhóm quyền>
```

Trong đó:

+ Tên nhóm quyền: Là tên của nhóm quyền cần tạo và phải duy nhất trong một cơ sở dữ liệu.

Ví dụ 13. Tạo nhóm quyền có tên là Xem_dulieu.

```
sp_addrole 'Xem_dulieu'
```

– **Thêm quyền vào nhóm (trình bày ở phần quản lý quyền trên cơ sở dữ liệu)**

– **Thêm một user vào nhóm quyền:**

```
sp_addrolemember <tên nhóm quyền>, <tên người dùng>
```

Ví dụ 14. Thêm người dùng ThanhThuy vào nhóm quyền Xem_dulieu.

```
sp_addrolemember 'Xem_dulieu' , 'ThanhThuy'
```

– **Xoá user khỏi nhóm quyền:**

```
sp_droprolemember <tên nhóm quyền>,<tên người dùng>
```

Ví dụ 15. Xoá người dùng ThanhThuy khỏi nhóm quyền Xem_dulieu

```
sp_droprolemember 'xem_dulieu','ThanhThuy'
```

– **Huỷ nhóm quyền:**

```
sp_droprole <tên nhóm quyền>
```

Ví dụ 16. Huỷ bỏ nhóm quyền có tên Xem_dulieu

```
sp_droprole Xem_dulieu
```

4.3.4. Quản lý quyền trên cơ sở dữ liệu

Có 2 loại quyền: Quyền thao tác trên các đối tượng cơ sở dữ liệu và quyền định nghĩa các đối tượng trong cơ sở dữ liệu.

4.3.4.1. Quyền thao tác trên đối tượng (Object Permissions)

Quyền thao tác trên đối tượng trong cơ sở dữ liệu là các quyền được cấp phát để người sử dụng có thể truy cập các đối tượng như bảng (table), bảng ảo (View) hay thủ tục (stored procedure), hàm (function) bằng các lệnh sau đây:

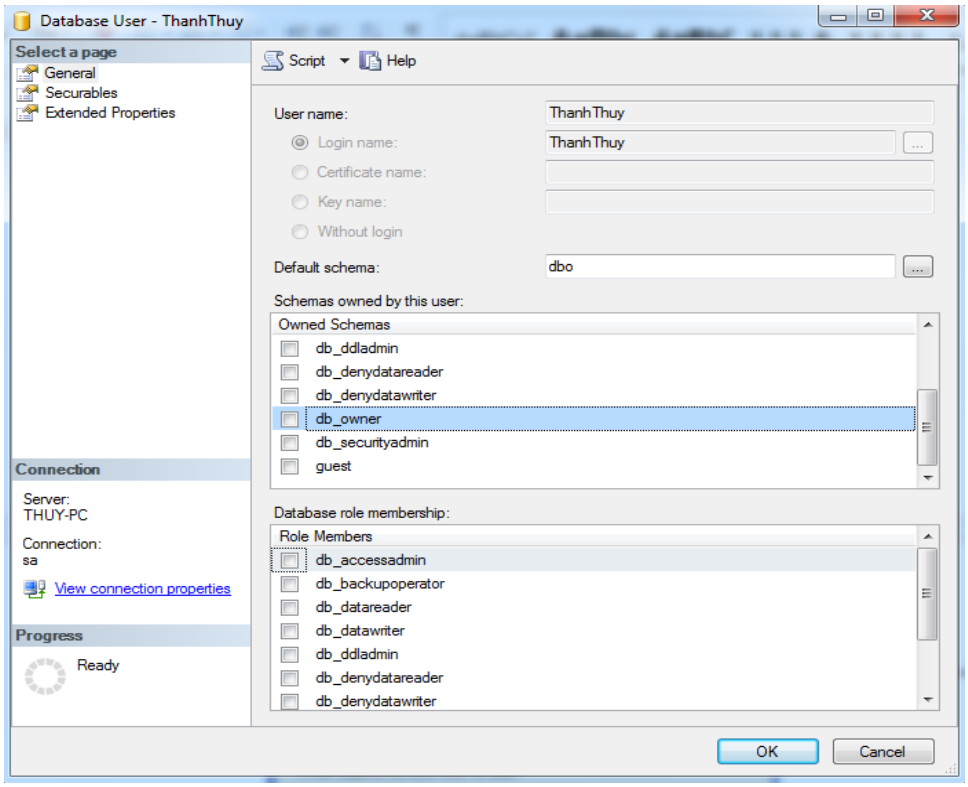
Bảng 4. 3. Nhóm quyền thao tác trên đối tượng:

Quyền	Diễn giải
SELECT	Cho phép người dùng nhìn thấy dữ liệu, có quyền thực thi câu lệnh SELECT trên bảng hay View.
INSERT	Cho phép người dùng thực thi câu lệnh INSERT
UPDATE	Cho phép người dùng thực thi câu lệnh UPDATE
DELETE	Cho phép người dùng thực thi câu lệnh DELETE
REFERENCES	Cho phép người dùng thêm dữ liệu vào bảng có khóa ngoại bằng câu lệnh INSERT.
EXECUTE	Cho phép người sử dụng thực thi các thủ tục.

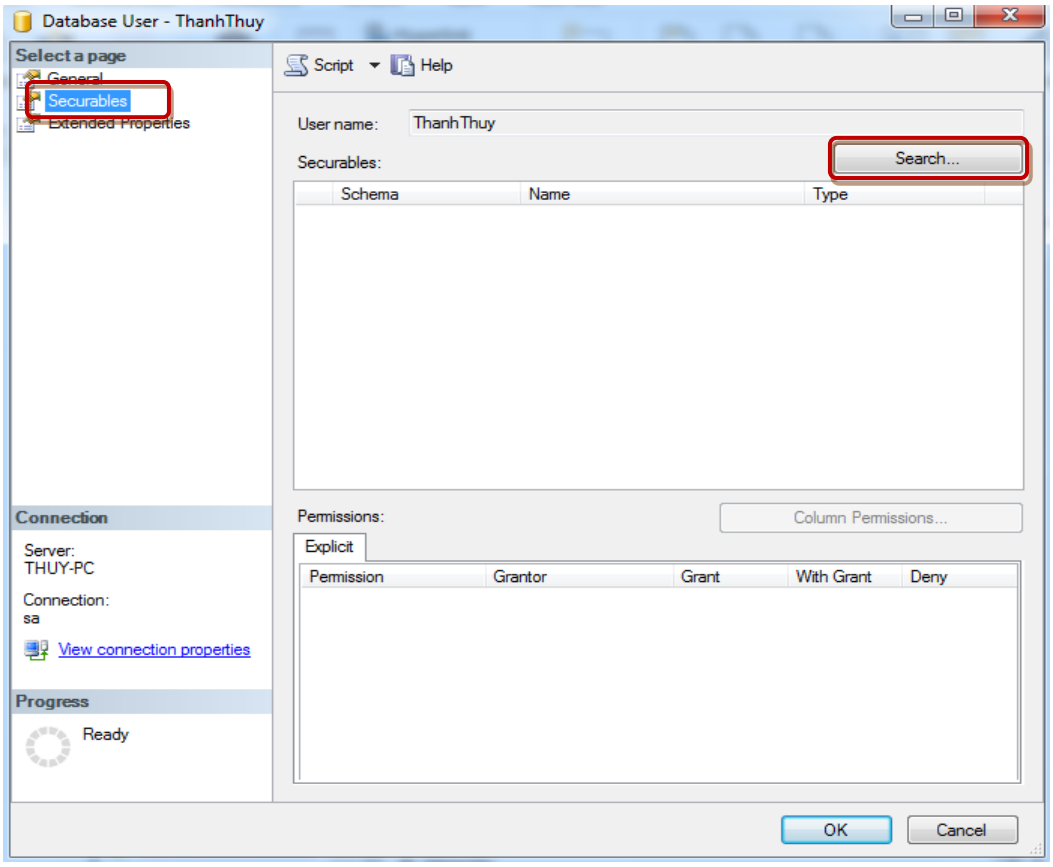
a. Cấp quyền

Cấp quyền cho user hoặc nhóm quyền do người dùng định nghĩa bằng cách sau:

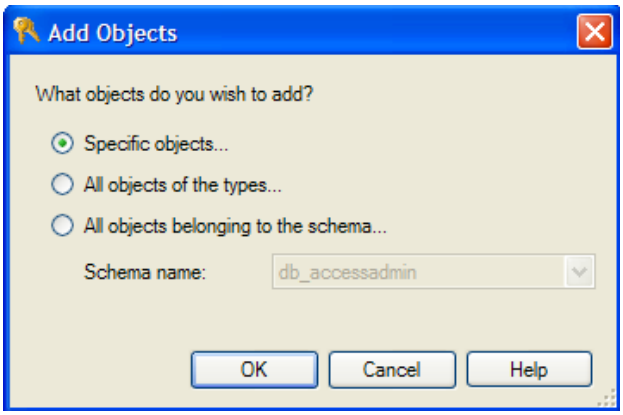
Nhấp chuột phải vào tên user / tên nhóm quyền → chọn Properties xuất hiện cửa sổ như sau:



Trong mục Select a page, chọn Securables:



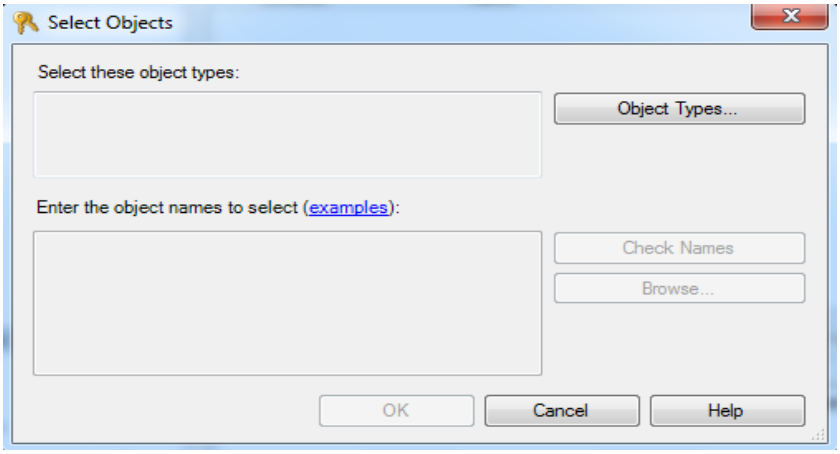
Click vào nút Search … , xuất hiện hộp thoại sau:



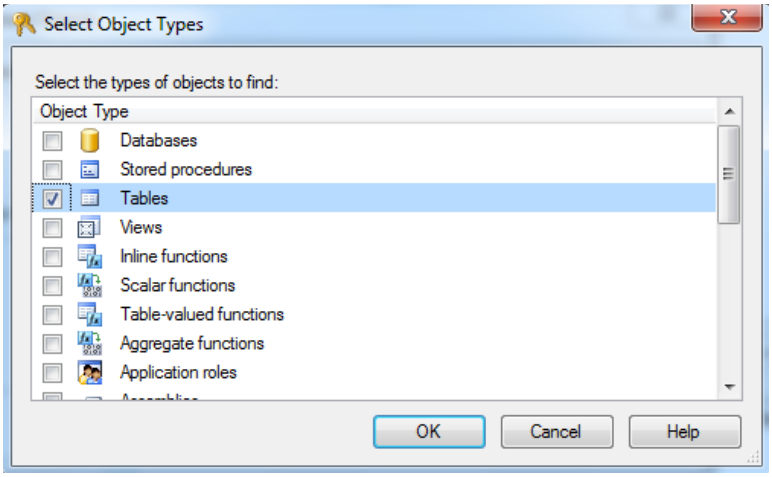
Trên cửa sổ add objects có 3 lựa chọn:

- Specific objects: các đối tượng cụ thể
- All objects of the types: các đối tượng của kiểu cụ thể
- All objects belonging to schema…: các đối tượng thuộc giản đồ

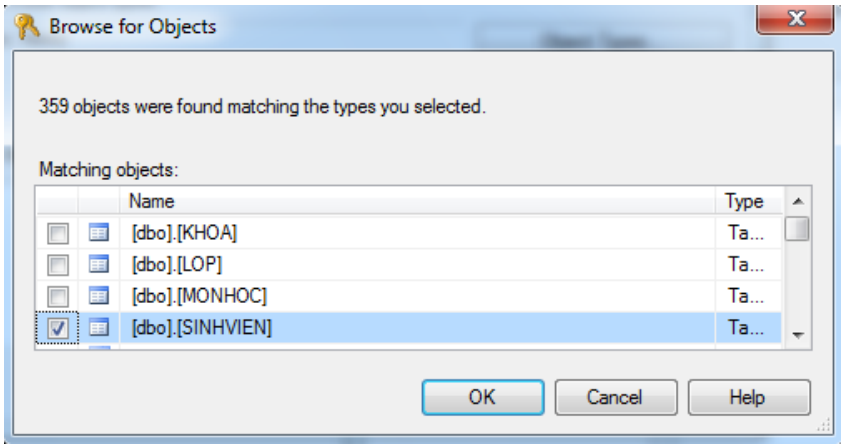
Chọn Specific objects → nhấn OK, xuất hiện hộp thoại sau:



Click vào nút Object Type, hộp thoại Select Object Types xuất hiện, ta chọn các kiểu đối tượng muốn bảo mật và click OK :

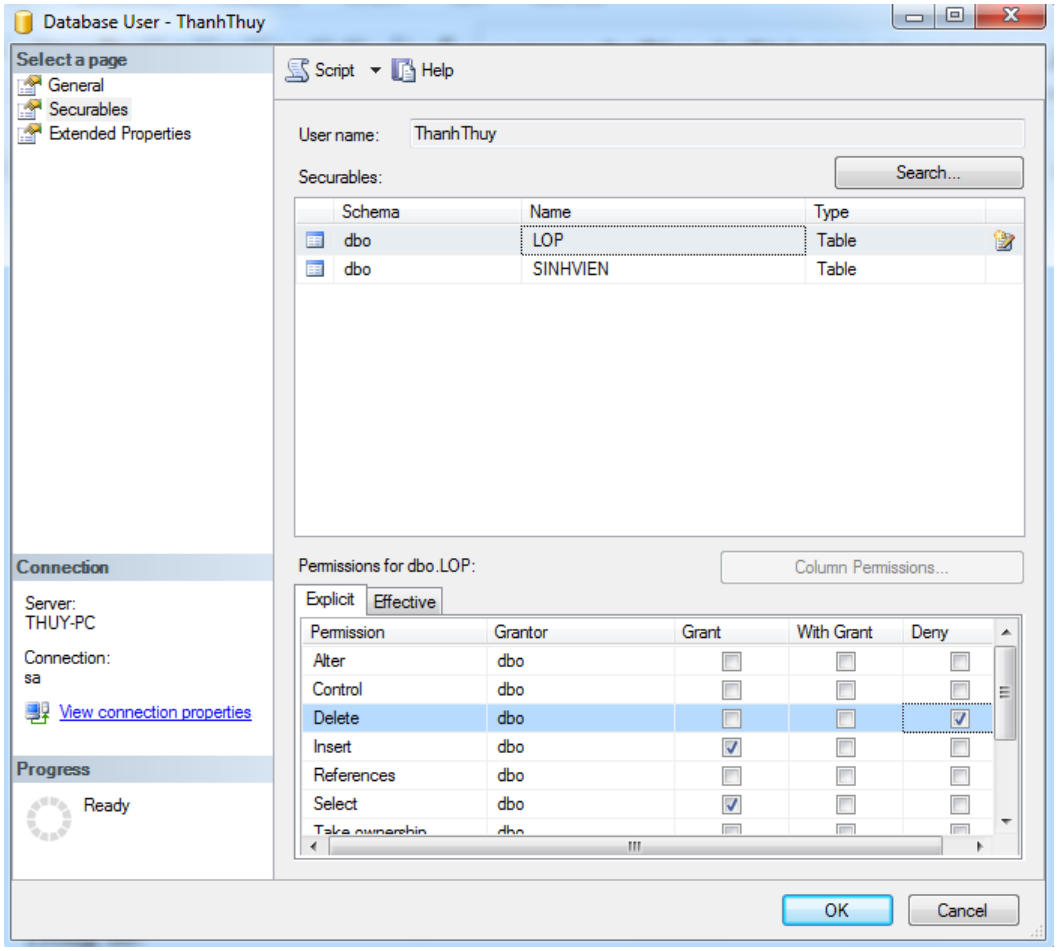


Click vào nút Browse … trên hộp thoại Select Objects, chọn các đối tượng cụ thể cần cấp quyền, sau đó click OK



Quay trở lại cửa sổ ban đầu, cấp quyền cụ thể cho từng đối tượng:

- Grant: Cấp quyền
- With Grant: Cho phép người dùng hoặc nhóm cấp quyền cho người dùng hoặc nhóm khác
- Deny: Thu hồi quyền



Ngoài ra, chúng ta có thể cấp quyền thao tác cho người dùng bằng lệnh T-SQL như sau:

Cú pháp:

```
GRANT  các_quyền_cấp_phát
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
|ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
|ON tên_thủ_tục
|ON tên_hàm
TO danh_sách_người_dùng | nhóm_người_dùng
[WITH GRANT OPTION ]
[AS role]
```

Trong đó:

- Quyền: là các lệnh SELECT, INSERT, UPDATE, DELETE, …
- Từ khoá ALL là cấp tất cả các quyền trên đối tượng liệt kê trong từ khoá ON bao gồm các quyền: select, insert, delete, update, references, execute.
- Tên đối tượng: Là tên đối tượng cần cấp quyền cho người dùng như table, view hay procedure.
- Tên user/nhóm quyền: Là tên người dùng hoặc tên nhóm quyền cần cấp quyền.
- Tuỳ chọn WITH GRANT OPTION: khi ta cấp phát quyền nào đó cho một người dùng trên một đối tượng cơ sở dữ liệu, người dùng đó có thể thực thi câu lệnh được cho phép trên đối tượng. Tuy nhiên người dùng đó lại không được cấp phát những quyền mà mình được phép cho những người dùng khác. Câu lệnh WITH GRANT OPTION cho phép người dùng đó chuyển tiếp quyền được cấp cho người dùng khác.
- Tùy chọn AS Role được chỉ định khi lệnh cấp quyền được thực hiện với tư cách là thành viên của role.

Ví dụ 17. Trên database QLSV, cấp phát cho người dùng có tên MinhTuan quyền thực thi các câu lệnh SELECT, INSERT và UPDATE trên bảng LOP

```
Use QLSV
Go
```

```
GRANT SELECT, INSERT, UPDATE
ON LOP
TO MinhTuan
```

Ví dụ 18. Cấp cho người dùng MinhTuan quyền xem họ tên và ngày sinh của các sinh viên (cột HOTEN và NGAYSINH của bảng SINHVIEN) trên database QLSV

```
GRANT SELECT(hoten,ngaysinh)
ON SINHVIEN
TO MinhTuan
```

Ví dụ 19. Cho phép người dùng *minhtuan* quyền xem dữ liệu trên bảng SINHVIEN đồng thời có thể chuyển tiếp quyền này cho người dùng khác

```
GRANT SELECT
ON SINHVIEN
TO MinhTuan
WITH GRANT OPTION
```

Ví dụ 20. Cấp các quyền SELECT trên bảng KETQUA cho nhóm quyền SV trên database QLSV, đồng thời có thể chuyển tiếp quyền này cho người dùng khác.

```
GRANT SELECT
ON KETQUA
TO SV
WITH GRANT OPTION
```

Giả sử người dùng ThanhThuy thuộc nhóm quyền SV muốn chuyển tiếp quyền select trên bảng KETQUA cho người dùng MinhTuan, ta thực hiện như sau:

```
GRANT SELECT
ON KETQUA
TO MinhTuan
AS SV
```

b. Thu hồi quyền đã cấp

Để thu hồi quyền đã cấp chúng ta sử dụng lệnh REVOKE có cấu trúc như sau:

Cú pháp:

```
REVOKE [GRANT OPTION FOR]
các_quyền_cần_thu_hồi [(danh_sách_cột)]|ALL
ON tên_bảng|tên_khung_nhìn
|ON tên_bảng|tên_khung_nhìn [(danh_sách_cột)]
|ON tên_thủ_tục
|ON tên_hàm
FROM danh_sách_người_dùng
[CASCADE]
[AS Role]
```

Ví dụ 21. Thu hồi quyền thực thi lệnh INSERT trên bảng LOP đối với người dùng MinhTuan

```
REVOKE INSERT
ON LOP
FROM MinhTuan
```

Ví dụ 22. Chỉ thu hồi quyền select trên cột ngày sinh của bảng SINHVIEN đối với người dùng MinhTuan

```
REVOKE SELECT
ON SINHVIEN(NGAYSINH)
FROM MinhTuan
```

Chú ý:

- Chỉ được thu hồi những quyền mà ta đã cấp phát trước đó.
- Những quyền mà người dùng được cấp bởi những người dùng khác thì vẫn còn có hiệu lực.

Ví dụ 23. Người dùng ThanhThuy thực hiện lệnh sau để cấp phát quyền Select dữ liệu trên bảng SINHVIEN cho người dùng Chaly:

```
GRANT SELECT
ON SINHVIEN
TO Chaly
```

Người dùng MinhTuan cấp phát quyền Select và Insert dữ liệu trên bảng SINHVIEN cho người dùng Chaly bằng câu lệnh:

```
GRANT SELECT, INSERT
ON SINHVIEN
TO Chaly
```

Sau đó, người dùng MinhTuan thu hồi quyền đã cấp cho người dùng Chaly bằng câu lệnh sau:

```
REVOKE SELECT, INSERT
ON SINHVIEN
FROM Chaly
```

Kết quả: Người dùng Chaly vẫn còn quyền select dữ liệu trên bảng SINHVIEN do người dùng ThanhThuy cấp.

– Nếu cấp phát quyền cho người dùng nào đó bằng câu lệnh GRANT với tùy chọn WITH GRANT OPTION thì khi thu hồi quyền bằng câu lệnh REVOKE phải chỉ định tuỳ chọn CASCADE. Trong trường hợp này, các quyền được chuyển tiếp cho những người dùng khác cũng đồng thời được thu hồi.

Ví dụ 24. Cấp phát quyền select trên bảng SINHVIEN cho người dùng MinhTuan như sau:

```
GRANT SELECT
ON SINHVIEN
TO MinhTuan
WITH GRANT OPTION
```

Sau đó người dùng MinhTuan cấp phát lại cho người dùng ThanhThuy quyền xem dữ liệu trên SINHVIEN với câu lệnh:

```
GRANT SELECT
ON SINHVIEN
TO ThanhThuy
```

Khi thu hồi quyền đã cấp phát cho người dùng MinhTuan, ta sử dụng câu lệnh REVOKE như sau:

```
REVOKE SELECT
ON SINHVIEN
FROM MinhTuan
CASCADE
```

Kết quả: Câu lệnh trên sẽ thu hồi quyền select trên bảng SINHVIEN của người dùng MinhTuan, đồng thời cũng thu hồi luôn quyền chuyển tiếp mà người dùng MinhTuan đã cấp cho người dùng ThanhThuy. Kết quả, cả người dùng MinhTuan và người dùng ThanhThuy đều không thể xem được dữ liệu trên bảng SINHVIEN.

– Khi cần thu hồi các quyền đã được chuyển tiếp và khả năng chuyển tiếp các quyền đối với những người đã được cấp phát quyền với tuỳ chọn WITH GRANT OPTION, trong câu lệnh REVOKE ta chỉ định mệnh đề GRANT OPTION FOR.

Ví dụ 25. Câu lệnh sau đây chỉ thu hồi quyền chuyển tiếp mà không thu hồi quyền select trên bảng SINHVIEN của người dùng MinhTuan.

```
REVOKE GRANT OPTION FOR SELECT
ON SINHVIEN
FROM MinhTuan
CASCADE
```

Kết quả:

+ Người dùng ThanhThuy sẽ không còn quyền xem dữ liệu trên bảng SINHVIEN
+ Người dùng MinhTuan không thể chuyển tiếp quyền đã được cấp phát cho những người dùng khác (tuy nhiên MinhTuan vẫn còn quyền xem dữ liệu trên bảng SINHVIEN).

Ví dụ 26. Giả sử ta đã cấp quyền SELECT trên bảng KETQUA cho nhóm quyền SV trên database QLSV, đồng thời cho phép các người dùng của nhóm quyền này chuyển tiếp quyền cho người dùng khác bằng câu lệnh sau:

```
GRANT SELECT
ON KETQUA
TO SV
WITH GRANT OPTION
```

Người dùng ThanhThuy thuộc nhóm quyền SV chuyển tiếp quyền select này cho người dùng MinhTuan bằng câu lệnh sau:

```
GRANT SELECT
ON KETQUA
TO MinhTuan
AS SV
```

Người dùng ThanhThuy có thể thu hồi lại quyền đã cấp cho người dùng MinhTuan bằng câu lệnh sau:

```
REVOKE SELECT
ON KETQUA
FROM MinhTuan
AS SV
```

c. Từ chối quyền

Để ngăn cản người dùng thực hiện các thao tác trên các đối tượng, ta sử dụng lệnh DENY có cấu trúc như sau:

Cú pháp:

DENY *các_quyền_từ_chối*
[(*danh_sách_cột*)] ON *tên_bảng* | *tên_khung_nhìn*
|ON *tên_bảng* | *tên_khung_nhìn* [(*danh_sách_cột*)]
|ON tên_*thủ_tục*
|ON *tên_hàm*
TO *danh_sách_người_dùng*
[CASCADE]
[AS Role]

Khi một user/role bị từ chối quyền, hệ thống sẽ ngăn không cho user sử dụng quyền và không cho phép user được thừa hưởng quyền này dù là thành viên của một role có quyền đó.

Nếu người dùng được cấp quyền với tùy chọn WITH GRANT OPTION thì phải chỉ định CASCADE khi thực hiện lệnh DENY.

Ví dụ 27. Ta cấp quyền select trên bảng SINHVIEN cho người dùng ThanhThuy và nhóm quyền SV bằng câu lệnh sau:

```
GRANT SELECT
ON SINHVIEN
TO ThanhThuy, SV
```

Thêm người dùng ThanhThuy vào nhóm quyền SV bằng câu lệnh sau:

```
sp_addrolemember 'SV' , 'ThanhThuy'
```

Ta tiếp tục thực hiện lệnh sau để thu hồi quyền select trên bảng SINHVIEN đã cấp cho người dùng ThanhThuy:

```
REVOKE SELECT
ON SINHVIEN
FROM ThanhThuy
```

Sau câu lệnh trên, người dùng ThanhThuy thực hiện được câu lệnh select trên bảng SINHVIEN do vẫn còn thừa hưởng quyền này từ role SV.

Muốn cho người dùng ThanhThuy không thể select trên bảng SINHVIEN và cũng không được thừa hưởng quyền này từ role SV, ta thực hiện câu lệnh sau:

```
DENY SELECT
ON SINHVIEN
TO ThanhThuy
```

Chú ý: Nếu user/role đang bị deny một quyền, lệnh REVOKE quyền này trên user/role sẽ gỡ bỏ đi hiệu lực của lệnh DENY đó.

Trong ví dụ trên, user ThanhThuy đang bị từ chối quyền Select trên bảng SINHVIEN, câu lệnh sau sẽ hủy bỏ đi hiệu lực của lệnh DENY và user ThanhThuy lại có thể thực hiện lệnh Select trên bảng SINHVIEN.

```
REVOKE SELECT
ON SINHVIEN
FROM ThanhThuy
```

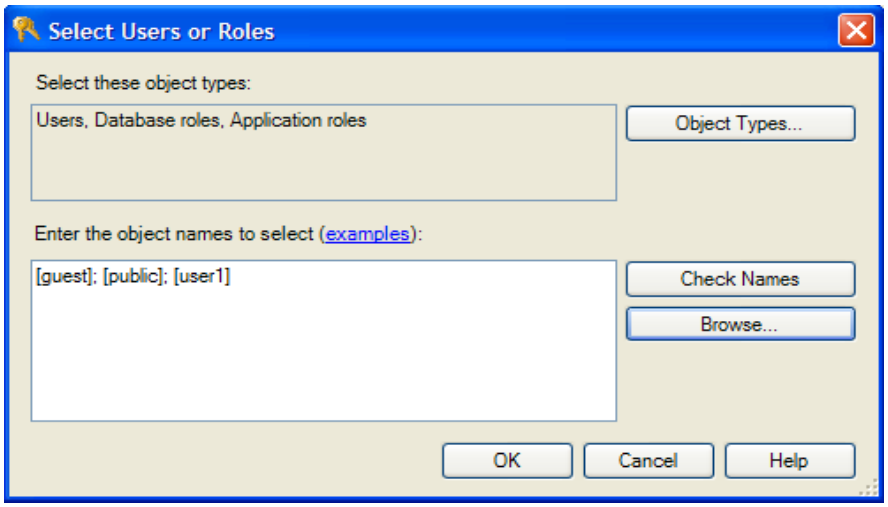
4.3.4.2. Quyền định nghĩa đối tượng

Quyền định nghĩa các đối tượng cho phép người dùng tạo ra các đối tượng cơ sở dữ liệu. Các quyền này thường là:

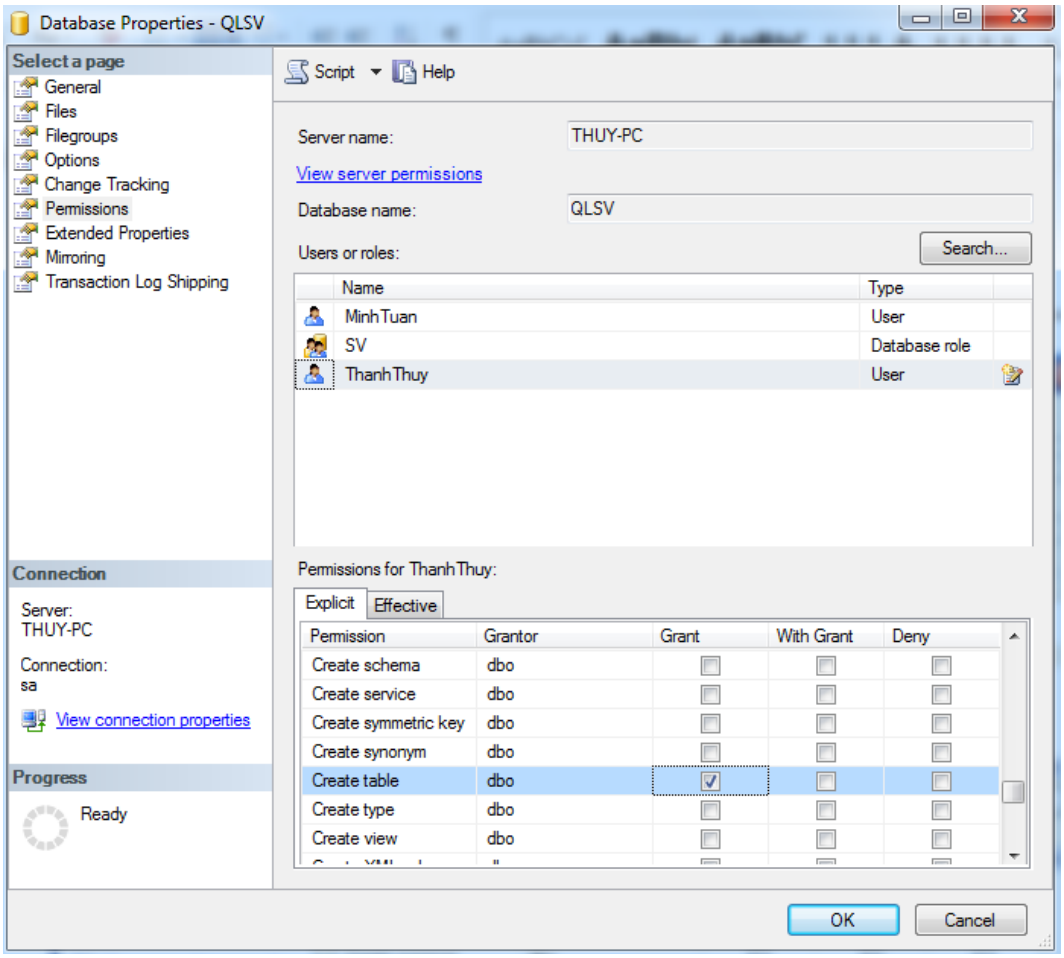
- Create Table: Cho phép người dùng tạo bảng.
- Create View: Cho phép người dùng tạo view.
- Create SP: Cho phép người dùng tạo stored procedure.
- Create Default: Cho phép người dùng tạo giá trị mặc định gắn với các cột trên bảng.
- Create Rule: Cho phép người dùng tạo ra các qui tắc.
- Create Function: Cho phép người dùng tạo ra các hàm do người dùng định nghĩa.
- Backup Database: Cho phép người dùng thực thi lệnh backup.
- Backup Log: Cho phép người dùng thực thi lệnh backup log.

Để tạo quyền định nghĩa các đối tượng, thực hiện như sau:

Click chuột phải vào cơ sở dữ liệu cần cấp quyền → Chọn Properties → chọn mục Permission → nhấn nút Search xuất hiện bảng sau:



Trên bảng Select Users or Roles chọn loại đối tượng (object Types) và nhấn nút Browse để chọn đối tượng cụ thể cần bảo mật.



Ngoài ra chúng ta cũng có thể dùng lệnh T-SQL để cấp quyền định nghĩa đối tượng với cú pháp như sau:

Cú pháp:

```
GRANT <các quyền định nghĩa đối tượng> | <ALL>
TO <tên user>
```

Trong đó:

- Các quyền định nghĩa đối tượng: Là các quyền định nghĩa đối tượng CSDL như đã trình bày ở phần trên.
- ALL: Cấp tất cả các quyền cho người dùng.
- Tên user: Là tên người dùng cần cấp quyền.

Chú ý:

- Với câu lệnh GRANT, ta có thể cho phép người sử dụng tạo các đối tượng trong cơ sở dữ liệu. Đối tượng cơ sở dữ liệu do người dùng nào tạo ra sẽ do người đó sở hữu và do đó người này có quyền cho người dùng khác sử dụng đối tượng và cũng có thể xóa bỏ (DROP) đối tượng do mình tạo ra.
- Câu lệnh GRANT trong trường hợp này không thể sử dụng tuỳ chọn WITH GRANT OPTION, tức là người dùng không thể chuyển tiếp được các quyền thực thi các câu lệnh đã được cấp phát.

Ví dụ 28. Cấp quyền Create Table cho người dùng có tên là ThanhThuy

```
GRANT Create Table
TO ThanhThuy
```

Bên cạnh việc cấp quyền thì chúng ta có thể thu hồi quyền đã cấp bằng lệnh REVOKE có cấu trúc như sau:

Cú pháp:

```
REVOKE <các quyền định nghĩa đối tượng> | <ALL>
FROM <tên user>
```

Ví dụ 29. Thu hồi quyền Create Table đã cấp cho USER1

```
REVOKE Create Table
FROM ThanhThuy
```

Kết chương

Sao lưu và phục hồi cơ sở dữ liệu là nhiệm vụ rất quan trọng, là mối quan tâm hàng đầu đối với các nhà quản trị cơ sở dữ liệu vì nó liên quan trực tiếp đến dữ liệu. Để quản trị một cách an toàn cần phải lập kế hoạch định kỳ sao lưu cơ sở dữ liệu theo một cách nào đó đề phòng rủi ro mất dữ liệu khi có sự cố xảy ra. Một nhiệm vụ quan

trọng khác là quản trị người dùng thông qua tạo, huỷ tài khoản đăng nhập, tài khoản người dùng, cấp quyền, huỷ quyền, quản lý thông qua nhóm quyền.

Câu hỏi và bài tập chương 4

Bài tập 1:

Cho cơ sở dữ liệu QLSV gồm bảng KHOA(Makhoa, Tenkhoa)

Chọn mô hình phục hồi là Full Recovery

Tạo kịch bản sao lưu gồm các bước như sau:

- Thêm dòng dữ liệu sau vào bảng Khoa:

Makhoa	Tenkhoa
CNTP	Công nghệ thực phẩm

- Thực hiện Full backup, tên file backup là FullQLSV.bak
- Thêm tiếp dòng dữ liệu sau vào bảng Khoa:

Makhoa	Tenkhoa
CNTT	Công nghệ thông tin

- Thực hiện Differential backup, tên file backup là DiffQLSV.bak
- Thêm tiếp dòng dữ liệu sau vào bảng Khoa:

Makhoa	Tenkhoa
CNHH	Công nghệ hóa học

- Thực hiện Log backup, tên file backup là LogQLSV.bak
- Thêm tiếp dòng dữ liệu sau vào bảng Khoa:

Makhoa	Tenkhoa
CNSH	Công nghệ sinh học

- Thực hiện Log backup, vào cùng file backup là LogQLSV.bak đã tạo ở trên.

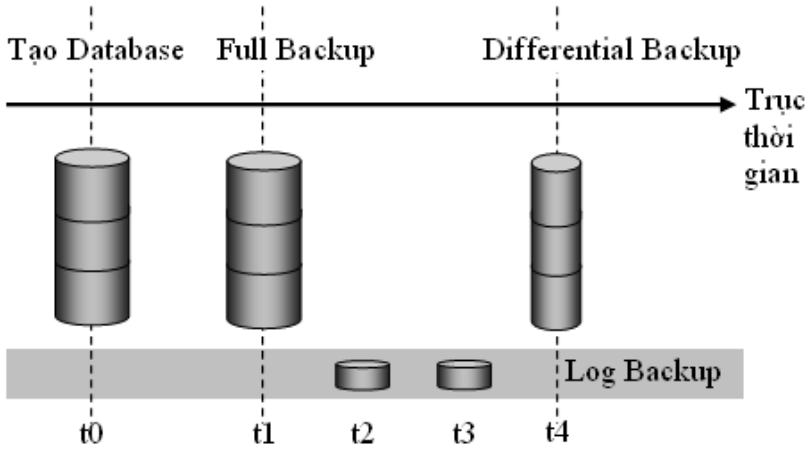
Giả sử lúc này hệ thống xảy ra sự cố, viết lệnh phục hồi database trên.

Bài tập 2:

a/ Tạo cơ sở dữ liệu gồm bảng SINHVIEN có nội dung như sau:

MASV	HOTEN	NGSINH	DCHI	LOP
3301090015	Tran Van Dang	11/06/1990	TPHCM	03CDNTH
3301090015	Nguyen Thi Lan	16/07/1991	HN	03CDNTH

b/ Dùng lệnh T-SQL sao lưu cơ sở dữ liệu trên theo lịch trình cho bởi hình vẽ sau:



Lưu ý: Tại mỗi thời điểm t_i ($i \geq 1$) sinh viên tự thêm một dòng dữ liệu vào bảng SINHVIEN để đảm bảo có sự thay đổi dữ liệu trong cơ sở dữ liệu.

c/ Viết lệnh phục hồi cơ sở dữ liệu trên tại thời điểm t3

Bài tập 3:

a/ Giả sử database QL_BANHANG được backup theo lịch trình sau:

Thời điểm sao lưu	Sự kiện
t1 (lúc 5pm thứ 7)	Full Backup
t2 (lúc 5pm thứ 2)	Log Backup
t3 (lúc 5pm thứ 4)	Differential Backup
t4 (lúc 5pm thứ 6)	Log Backup

Tại mỗi thời điểm t_i ($i \geq 1$) sinh viên tự thêm một dòng dữ liệu vào bảng HOADON để đảm bảo có sự thay đổi dữ liệu trong cơ sở dữ liệu và thực hiện backup bằng lệnh.

b/ Giả sử sự cố xảy ra lúc 8 giờ sáng thứ 7. Viết lệnh phục hồi cơ sở dữ liệu trên.

Bài tập 4:

a/ Giả sử database TUYENSINH được backup theo lịch trình sau:

Thời điểm sao lưu	Sự kiện
t1 (lúc 5pm thứ 7)	Full Backup
t2 (lúc 12am thứ 2)	Log Backup
t3 (lúc 5pm thứ 3)	Differential Backup
t4 (lúc 12am thứ 4)	Log Backup

Tại mỗi thời điểm t_i ($i \geq 1$) sinh viên tự thêm một dòng dữ liệu vào bảng THISINH để đảm bảo có sự thay đổi dữ liệu trong cơ sở dữ liệu và thực hiện backup bằng lệnh.

b/ Giả sử sự cố xảy ra lúc 4 giờ chiều thứ 4. Viết lệnh phục hồi cơ sở dữ liệu trên.

Bài tập 5:

Cho cơ sở dữ liệu QLSV gồm:

KHOA(Makhoa, Tenkhoa)

LOP(Malop, Tenlop, Makhoa)

SINHVIEN(MaSV, Hoten, Ngaysinh, Diachi, Gioitinh, Malop)

MONHOC(MaMH, TenMH, SoTC)

KETQUA(MaSV, MaMH, Diem)

1. Tạo và gán quyền các nhóm quyền sau:

Nhóm quyền	**Quyền**
Ban giám hiệu	Thêm, xóa, sửa dữ liệu bảng Khoa.
Đào tạo	Thêm, xóa, sửa dữ liệu bảng Lop, SinhVien, Ketqua
Khoa	Thêm, xóa, sửa dữ liệu bảng Monhoc
Sinh viên	Xem dữ liệu bảng Sinhvien và Ketqua

2. Tạo các tài khoản đăng nhập sau với chứng thực SQL:

Tên đăng nhập	**Mật khẩu**
Ngoan	abcxyz
Thanh	abcxyz

Nguyen	abcxyz
Phuong	abcxyz
Anh	abcxyz
Hoa	abcxyz
Hong	abcxyz
Dung	abcxyz

3. Tạo các user

Tên user	Tên đăng nhập	Thuộc nhóm quyền
Ngoan	Ngoan	Ban giám hiệu
Thanh	Thanh	Đào tạo
Nguyen	Nguyen	Đào tạo
Phuong	Phuong	Khoa
Anh	Anh	Khoa
Hoa	Hoa	Sinh viên
Hong	Hong	Sinh viên
Dung	Dung	Sinh viên

Bài tập 6:

Cho cơ sở dữ liệu QLBH gồm:

KHACHHANG(MaKH, Hoten, Diachi, DT)

MATHANG(MaMH, TenMH, DVT, SLton, Dongia)

HOADON(SoHD, Ngaylap, MaKH)

CHITIET_HD(SoHD, MaMH, SL)

Danh sách nhân sự trong công ty như sau:

Ban giám đốc:

Username	Password
NguyenA	abc
NguyenB	xyz

Phòng kế toán:

Username	Password
NguyenC	abc
NguyenD	xyz

Phòng kinh doanh:

Username	Password
NguyenE	abc
NguyenF	xyz

Phòng IT:

Username	Password
NguyenG	abc
NguyenH	xyz

Phân quyền cho các phòng ban trong công ty như sau:

- Ban giám đốc đươc phép xem toàn bộ cơ sở dữ liệu, có quyền chuyển tiếp quyền này cho user khác.
- Phòng kế toán được phép thêm, cập nhật trên bảng hóa đơn và chi tiết hóa đơn
- Phòng kinh doanh được phép thêm, và cập nhật trên bảng khách hàng
- Phòng IT được toàn quyền trên cơ sở dữ liệu
- Không cho user NguyenC quyền cập nhật trên bảng hóa đơn và chi tiết hóa đơn.
- Thu hồi quyền chuyển tiếp của ban giám đốc.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Thiên Bằng (2006), *Khám phá SQL Server 2005*, NXB Lao động Xã hội.

[2] Phạm Hữu Khang (2008), *SQL Server 2005 – Lập trình nâng cao*, NXB Lao động Xã hội,

[3] Phạm Hữu Khang (2008), *SQL server 2005 lập trình thủ tục và hàm*, NXB Lao động Xã hội.

[4] Phạm Hữu Khang (2007), *SQL server 2005 lập trình T-SQL*, NXB Lao động Xã hội.

[5] Michael Otey, Denielle Otey (2006), *Microsoft® SQL Server™ 2005 Developer's Guide*, The McGraw-Hill Companies.

[5] Dan Wood, Chris Leiter, Paul Turley (2007), *Beginning SQL Server™ 2005 Administration*, Wiley Publishing, ISBN-13: 978-0-470-04704-0.